Feedback Circuit for Organic LED Active-Matrix Display Drivers

by

Eko T Lisuwandi

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
May 10, 2002

Author_____

Department of Electrical Engineering and Computer Science
May 10, 2002

Certified by_____

Charles G. Sodini
Thesis Supervisor

Certified by_____

Vladimir Bulovic
Thesis Supervisor

Accepted by_____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Feedback Circuit for Organic LED Active-Matrix Display Drivers

by

Eko T Lisuwandi

Submitted to the
Department of Electrical Engineering and Computer Science

May 10, 2002

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

A feedback circuit for an Organic Light Emitting Diode (OLED) based display is proposed and demonstrated. An OLED-based flat panel display is brighter, much lower power, has no viewing angle limitation and potentially cheaper compared to available Liquid Crystal (LC) based displays. Despite these advantages, an OLED-based display is not widely commercialized mainly due to its short practical lifetime. The I-V characteristics of the individual OLED pixels vary over time, temperature and processing-dependent parameters. Moreover, the variation is not uniform across an array of OLED pixels, causing OLED based displays to lose brightness accuracy after a few thousand hours of operation. The proposed feedback circuit is used to compensate for the non-uniformities in the individual OLED characteristics. The resulting display leverages the beneficial aspects of OLED display technology, while maintaining pixel uniformity and grayscale reproducibility. A demonstration system is built proving the feasibility of a flat panel display using direct optical feedback. The feedback loop monitors the output light level using a sensor and adjusts the current fed to the pixels to set the output light power to a digitally set reference level. The system shares a single feedback loop among a number of pixels, saving power and real estate. The demonstration system consists of a 5x5 array of LEDs, a CMOS camera, analog pixel circuitry, driver and feedback loop, as well as a digital controller. The demonstration system also shows the feasibility of time-sharing a feedback loop among a number of output devices.

Thesis Supervisors:

1. Charles G. Sodini

   Professor of Electrical Engineering
2. Vladimir Bulovic

   Assistant Professor of Electrical Engineering

Acknowledgements

I am very grateful to the number of people who have played a part in the successful completion of this master thesis project.

First, I would like to extend my gratitude to Professor Charles Sodini and Professor Vladimir Bulovic for allowing me this research opportunity. Throughout the project both of them exhibited trust in my ideas and confidence in my ability. Special thanks to Professor Bulovic for some of the figures included in this thesis and several of the presentations related to this thesis.

I thank all of the students and staff in Professor Sodini's research group. Many thanks to Steven Decker and Keith Fife who have developed the digital camera and the camera processing board used in this thesis. Souren Lefian offered assistance and advice with design and general engineering problems. Many thanks also to Patricia Varley and Kathy Patenaud for their assistance in dealing with MIT administrative procedures.

Finally, I thank my friends and family. I thank my girlfriend, Prima Fortunadewi, for her continuing support throughout the project. My Mom and Dad for the never-ending encouragement for me to realize my potential, and to explore my electronics interest while growing up. I am very grateful for their constant moral and monetary support.

Contents

1. Introduction **8**

1.1. Liquid Crystal Display Limitations 8

1.2. Organic Light Emitting Device as a Promising Alternative 9

2. Background and Motivation **11**

2.1. Organic LED Structure and Operation 11

2.2. Electrical Characteristics Variation of OLED 13

2.2.1. Power Efficiency Degradation and Power Characteristic Variation 13

2.2.2. I-V Characteristic Variation 15

2.2.3. Previous Works on OLED-Based Flat Panel Displays 16

3. Feedback Ideas and Transparent OLED **17**

3.1. Transparent OLED 17

3.2. Direct Optical Feedback 18

3.3. A Feedback Loop Shared by a Column of Pixels 19

4. System Implementation **21**

4.1. Preliminary Integrated Design 21

4.1.1. Feedback Network and Pixel Circuitry 21

4.1.2. Pixel Array 22

4.2. 5x5 Pixel Demonstration System 23

4.2.1. Feedback Network and Pixel Circuitry 23

4.2.1.1. Pixel Circuitry 24

4.2.1.2. The Feedback Loop 25

4.2.1.3. Compensating the Feedback Loop 25

4.2.2. Pixel Array 31

4.2.3. Digital Interface and Control 32

4.2.3.1. The Digital Camera 32

4.2.3.2. Digital Control 33

4.2.3.2.1. Overview of the Digital Controller 33

4.2.3.2.2. The State Machine and The Modules 35

5. Results **39**
5.1. Proto Board Prototype 39
5.2. Printed Circuit Board 40
5.3. Display Uniformity with Feedback 40
6. Conclusion **42**
6.1. Summary 42
6.2. Future Work 42
7. References **44**
8. Appendix A **46**
9. Appendix B **60**

List of Figures

1. The Basic Physical Structure of an LCD cell 9
2. Typical Structure of an Organic LED and the Molecular Structure of Alq_3 and TPD 11
3. Energy Band Illustration of Optical Recombination in an Organic LED 12
4. (a) Average Optical Power Variation of 10 OLEDs with Time and Use 14
   (b) Average Voltage Variations Required Maintaining a Current of 1mA through the OLEDs 14
5. I-V Characteristic Variation with Temperature 15
6. I-V Characteristic Variation with Thickness of the Organic EL Layer 15
7. An Alphanumeric Display Made from Transparent OLEDs 17
8. Transparent OLED on a Silicon Photodetector 18
9. The Block Diagram of Main Feedback Idea 18
10. Simple Schematic for the Shared Feedback Loop Idea 20
11. Integrated Single Pixel Feedback Loop Schematic Details 22
12. The Schematic of an Integrated 5x5 Pixels Array 22
13. The Overall System Block Diagram 23
14. Demonstration System Feedback Loop Schematic 23
15. Pixel Implementation Schematic Details 24
16. Block Diagram of the Feedback Loop 25
17. Reference Loop Used to Estimate Step Response 26
18. (a) Loop Transmission Bode Plot with Pure Time Delay 27
    (b) Loop Transmission Bode Plot with PADE Approximation 27
19. Step Response of the Closed Loop System with $\phi_m=60^o$ 27
20. Step Response of the Actual Closed Loop System with R=820KΩ and C=0.33uF at Maximum Brightness Operating Point 28
21. Step Response of the Actual Closed Loop System with R=820KΩ and C=0.33uF at Minimum Brightness Operating Point 29
22. Large Signal Step Response 29
23. Demonstration System Feedback Loop Schematic with Component Types and Values 30
24. The Simplified Schematic of the 5x5 Pixels Array with the Driving Circuit 31

25. Output Signals from Camera 32
26. Overall Functional Block Diagram of the Digital Controller 33
27. Detailed Functional Schematic of the Digital Controller 34
28. State Diagram for the Digital Controller State Machine 35
29. (a) The Proto Board Implementation of the System 39
(b) The Details of the Analog Driver 39
30. Final Demonstration System Printed Circuit Board 40
31. (a) Pixel Array Showing Non Uniformity in Brightness 41
(b) The Same Pixel Array Showing Uniform Brightness After Feedback Correction 41
32. (a) An Object Placed in Between Camera and LEDs Array Covering Some Pixels 41
(b) The Same Array Immediately After Object is Removed 41
(c) The Same Array after the Feedback Correction 41

Chapter 1

Introduction

As the necessity for mobile computing grows, the demand for good quality, light and reliable displays has also increased. This is even more evident in light of the rapid decline in price, and increased availability of notebook/palmtop computers capable of supporting Graphical User Interface (GUI) Operating Systems (OS) such as Windows. The current display technology that satisfies all the requirements falls under the category of flat panel displays. The most popular of which is the Liquid Crystal Displays (LCD). Virtually all notebook computers nowadays are equipped with some form of LCD. However, LCD as a mature technology has many limitations. As a result there has been a lot of research to find an alternative. The most promising of which is Organic Light Emitting Device (OLED) Display.

1.1. Liquid Crystal Display Limitations

Most of the limitations of LCD technology come from the fact that LCD is a non-emissive display device. This means that they do not emit light on their own. Thus, an LCD operates on the basis of either passing or blocking light that is produced by an external light source (usually from a backside lighting system or reflecting ambient light). Applying an electric field across an LCD cell controls its transparency or reflectivity. A cell blocking (absorbing) light will thus be seen as black and a cell passing (reflecting) light will be seen as white. For a color displays, there are color filters added in front of each of the cells and a single pixel is represented by three cells, each responsible for the basic colors: red, green and blue.

The basic physical structure of an LCD cell is shown in Figure 1 [1]. The Liquid Crystal (LC) material is sandwiched between two polarizers and two glass plates (or between one glass plate and one Thin Film Transistor (TFT) layers). The polarizers are integral to the working of the cell. Note that the LC material is inherently a transparent material, but it has a property where its optical axis can be rotated by applying an electric field across the material. When the LC material optical axis is made to align with the two polarizers' axis, light will pass through the second polarizer. On the other hand, if the optical axis is rotated 90 degrees, light will be polarized by the first polarizer, rotated by the LC material and blocked by the second polarizer.

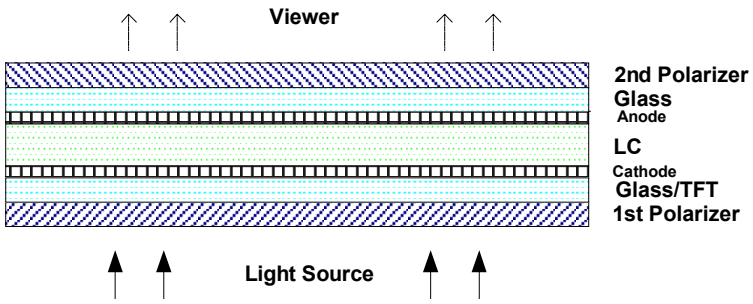


Figure 1. The Basic Physical Structure of an LCD cell

Note that the polarizers and the LC material absorb light. On a typical monochrome LCD display the polarizers alone absorb 50% of the incident light. On an active matrix display with a TFT layer, the light throughput may be as low as 5% of the incident light. Such low light output efficiency requires LC based displays to have a powerful backside or ambient light illumination to achieve sufficient brightness. This causes LCDs to be bulky and power hungry. The LC cells are in fact relatively thin and their operation relatively power efficient. It is the backside light that takes up most space as well as power. In fact with the advent of low power microprocessors, the LCD module is the primary cause of short battery life in notebook computers.

Moreover, the optical properties of the LC material and the polarizer also causes what is known as the viewing angle effect. The effect is such that when a user is not directly in front of the display, the image can disappear or sometime seem to invert (dark images become light and light images become dark).

1.2. Organic Light Emitting Device as a Promising Alternative

With these disadvantages of an LC based display in mind, there has been a lot of research to find an alternative. In recent years, a large effort has been concentrated on Organic Light Emitting Device (OLED) based displays. OLED-based displays have the potential of being lighter, thinner, brighter and much more power-efficient than LC based displays. Moreover, OLED-based displays do not suffer from the viewing angle effect.

Organic Optoelectronics has been an active field of research for nearly two decades. In this time device structures and materials have been optimized, yielding a robust technology. In fact, OLEDs have already been incorporated into several consumer electronic products. However, there are basic properties of organic molecules, especially their instability in air, that hamper the commercialization of the technology for high quality displays.

Previous efforts to increase the reliability and accuracy of the OLED light output has focused mainly on having predictable current level in an OLED device in the face of varying threshold voltages of the driving transistors. The effort is a direct parallel to the Thin Film Transistor technology used in Liquid Crystal Displays.

In this thesis, a feedback circuit for OLED based displays is proposed and demonstrated. The circuit monitors the light output of the pixels and uses this information to compensate for the non-uniformities in the individual OLED pixel characteristics over time, temperature and processing-dependent parameters. The resulting display leverages the beneficial aspects of organic light emitting diode (OLED) display technology, while maintaining pixels array uniformity and grayscale reproducibility.

The rest of this thesis is organized as follow: Chapter 2 discusses the specific background and motivation for this thesis related to the physical properties of the OLED; Chapter 3 discusses the idea of using direct optical feedback for OLED based displays; Chapter 4 presents the implementation details of the thesis project; Chapter 5 shows some of the result obtained using the demonstration system; and Chapter 6 ends the thesis with conclusions.

Chapter 2

Background and Motivation

The first part of this section discusses the physical structure of Organic LEDs and the basic operations of such devices. The second part discusses the electrical characteristics and limitations of OLEDs that motivate the research presented in this thesis.

2.1. Organic LED Structure and Operation

An Organic LED is a light emitting device whose p-n junction is made from an organic compound such as: Alq_3 (Aluminum tris (8-hydroxyquinoline)) and diamine (TPD). A typical structure of an OLED cell and the molecular structure of some typical organic materials used are shown in Figure 2:

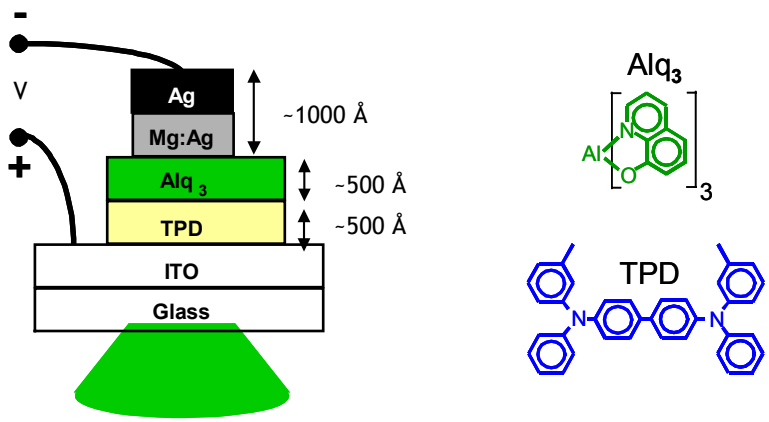


Figure 2. Typical Structure of an Organic LED and the Molecular Structure of Alq_3 and TPD

For an Organic LED, the organic layer corresponding to the p-type material is called the hole-transport layer (HTL) and similarly the layer corresponding to the n-type material is called the electron-transport layer (ETL). In Figure 2, Alq_3 is the ETL and TPD is the HTL. Similar to doped silicon, when ETL and HTL materials are placed to create a junction, the energy bands equilibrates to maintain continuity across the structure. When a potential difference is applied across the structure, a drift current flows through the structure. The injected carriers

recombination at the junction consists of both thermal and optical recombination, which emits photons.

Figure 3 shows the optical recombination from the energy band perspective. Note that LUMO is a short form for Lowest Unoccupied Molecular Orbital, which corresponds to the conduction band in the energy diagram of doped silicon, and HOMO is a short form for Highest Occupied Molecular Orbital, which corresponds to the valence band in the energy diagram of doped silicon.

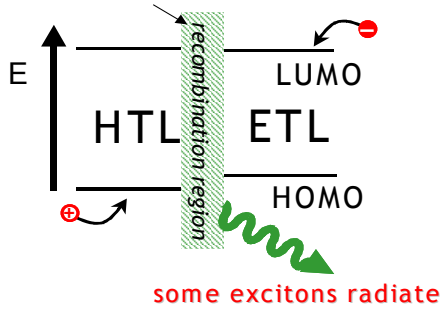


Figure 3. Energy Band Illustration of Optical Recombination in an Organic LED

Since an OLED emits light through a recombination process, it does not suffer from the viewing angle limitation like an LC based device. Note that for any device to become a viable candidate for use in flat panel displays it has to be able to demonstrate high brightness, good power efficiency, good color saturation and sufficient lifetime. Reasonable lower limits specifications [2] for any candidate device should include the following: brightness of ~ 100 cd/m^2, operating voltage of 5-15V and a continuous lifetime of at least 10,000h.

OLEDs with brightness of up to 140,000 cd/m^2 [3], power efficiencies of up to 40 lm/W [4], and low operating voltages from 3-10V [5] have been reported. Saturated-color OLEDs have been demonstrated, spanning almost the entire visible spectrum. Moreover, the thickness of an OLED structure, which typically is less than a micrometer, allows for mechanical flexibility, leading to the development of bendable displays [6] indicating the potential development of rolled or foldable displays. Furthermore, the recent development of vapor phase deposition techniques for the OLED manufacturing process may well result in low-cost large-scale production of OLED based flat panel displays as opposed to LC based displays that require extra processes such as layer alignment and tilt angle adjustment.

OLED lifetime exceeding 50,000 h [7] has been reported. Note however, this lifetime number applies to any singular OLED structure. The number does not capture the fact that each OLED pixel's electrical characteristics in a display consisting of array of pixels may vary

differently than the characteristic of its neighboring pixel. Although all the pixels in the array may have up to 6 years lifetime, a display consisting of pixels with differing characteristics will lose its brightness and pixel-to-pixel accuracy if no adjustments are made to compensate for this variation.

2.2. Electrical Characteristics Variation of OLED

With the advantages mentioned in section 2.1, one may wonder why OLED based displays are not as popular among consumer mobile computing devices as LC based displays. For one thing, building practical OLEDs had remained a challenge until 1987 when the first vacuum-deposited OLEDs with low operating voltages were made [8]. On the other hand, liquid crystal development has had a long history and by 1977 there was already a lot of research done on Twisted Neumatic LC based displays [9].

There are also other challenges in OLED based flat panel displays design, which are not found with LC based design. As mentioned in the previous section, OLED pixels in an array may not have uniform electrical characteristics since OLEDs are organic devices whose electrical properties are easily affected by their environment and its pattern of usage.

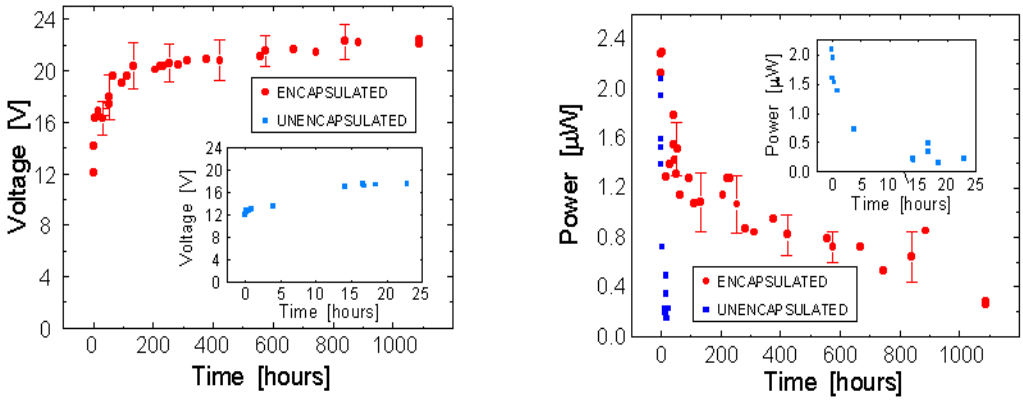
The most detrimental of these variations is the power efficiency degradation of the OLED with time and use. Although all the pixels in an array will have the same age but not all of them will have an identical pattern of usage. This causes differing degradation between one pixel and another or even between pixels of one color and another.

In addition to this, the initial I-V characteristics of OLED cells depend on a large number of factors, which may be difficult to control precisely.

2.2.1. Power Efficiency Degradation and Power Characteristic Variation

When a fixed current is supplied continuously to an OLED device, one would expect the same intensity of light to be emitted if the power efficiency and the I-V characteristics of the device remain constant. However, as reported by Burrows et al. (1994), when an OLED made using Alq_3 as the active Electro Luminescence (EL) layer is driven with current of $5mA/cm^2$, it lost 50% of its initial intensity in 100h even when it is operated in dry Ar (Argon) [10]. It was also reported that a blue OLED showed a decrease of 90% of its initial EL intensity in 130h when operated in dry N_2 (Nitrogen). Operating OLEDs in air resulted in 99% loss of EL intensity in as little as 150 min. It is indeed clear that such great output intensity variation during the device lifetime, even in inert atmospheres, render OLEDs ineffective for commercial applications.

In that particular paper by Burrows et al. (1994), an encapsulation mechanism is suggested that prolongs the lifetimes of the OLEDs device by two orders of magnitude. Even with the encapsulation mechanism, the power-efficiency characteristic of the device does not remain constant during its lifetime although it remains within an acceptable range for sufficient brightness. Figure 4 shows the average optical power output degradation over time and the voltage required to maintain the same level of current through the OLED device from early studies of OLED devices [10]. Present devices have the same characteristic curves although the lifetime is increased to the order of 10000-20000 hours.



(a) (b)

Figure 4. (a) Average Optical Power Variation of 10 OLEDs with Time and Use [10]
(b) Average Voltage Variations Required Maintaining a Current of 1mA through the OLEDs [10]

The plots in Figure 4(a) were obtained using the same constant level of current of 1mA as in Figure 4(b). Therefore, in addition to the power efficiency variation, the power output characteristics also change in a non-uniform way with time. A particular level of current and/or voltage does not correspond to a fixed level of optical power output. As such, an additional mechanism has to be introduced to control the optical power output level.

2.2.2. I-V characteristic variation

As can be inferred from the power characteristic variation, the I-V characteristic of an OLED is also varying with time. Several factors contribute to the I-V characteristic variation.

The first and foremost is temperature. As shown in Figure 5, the I-V characteristic depends quite strongly on the operating temperature [2]. Similar to the power characteristic variation, this I-V characteristic variation pose a challenge to the control of OLED based displays as the I-V operating points have to be shifted depending on the operating temperature.

Besides temperature, the I-V characteristic also depends strongly on the type of anode/cathode used in the device as well as the thickness of the organic active Electro Luminescence (EL) layer. In particular Figure 6 shows the I-V characteristic variation with the thickness of the organic layer [11].

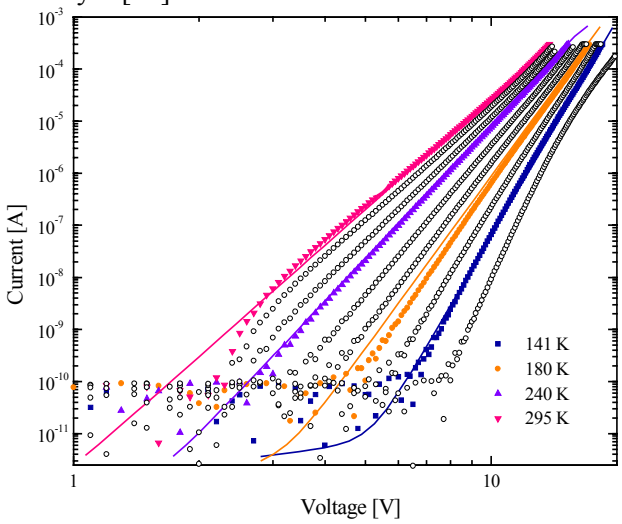


Figure 5. I-V characteristic Variation with Temperature [2]

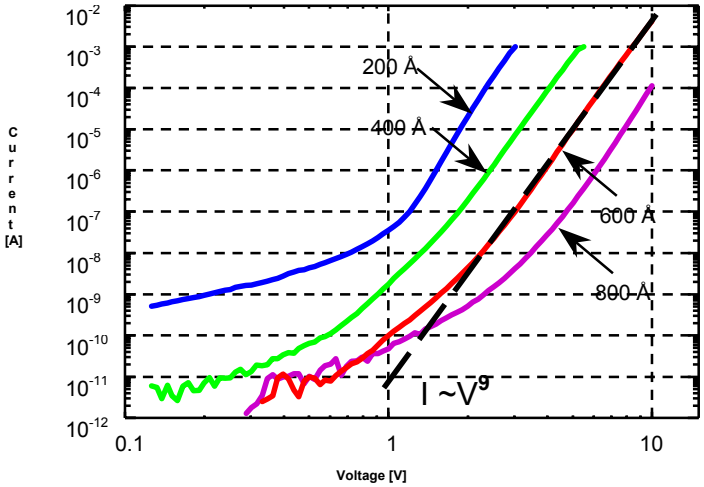


Figure 6. I-V characteristic Variation with Thickness of the Organic EL Layer [11]

Both the power and the I-V characteristics variation indicate the necessity of real time feedback control to be able to provide a consistent relation between the electrical input signal and optical power output. It is the main motivation of this thesis to be able to develop a feedback circuit that can be implemented to produce a reliable OLED-based flat panel display with a predictable optical power output.

2.2.3. Previous Works on OLED-Based Flat Panel Displays

There has been a lot of research done recently on the implementation of an OLED-based flat panel displays. Dawson et al. [12] have characterized the low current operating mode of OLED device. There have also been works done on other aspects of the OLED-based displays such as active matrix addressing [13] and pixel design [14,15].

Most of the work in the literature that deals with the pixel design and optical power-output control of the OLED cells has mostly been dealing with an effective way of producing a predictable current level in each device for an array of OLED cells, in the face of electrical characteristic variation of the Thin Film Transistor (TFT) that drives the cells.

The methods introduced by Dawson et al. [14] and Yi He et al. [15] include a mechanism in which the TFT can be actively programmed in real time so that it can drive a particular value of current into the OLED cells within any cycle. Both of the pixel designs that they proposed used 4 transistors. With their design, they have reported that they are able to demonstrate an array of OLED cells producing uniform optical power output.

However, none of the research done so far takes into account the OLED cells luminance efficiency variation and power output degradation with time. If an OLED is to be used in a commercial flat panel display application, it has to be able to be driven reliably with some kind of electric signal so that it produces a consistent optical power output for some defined electrical input signal during its lifetime.

When no provision is made for this OLED cell degradation, any OLED-based flat panel display will produce an output whose overall brightness degrades slowly, typically on the order of 10000h. Most important of all, the pattern of usage of each of the OLED cells in the display will not be identical over the display life time and without feedback, the whole display may be unusable once any one cell degrades relative to another. The time constant for this relative degradation may be as short as 100-1000h.

Chapter 3

Feedback Ideas and Transparent OLED

The limitations due to the variation in the OLED characteristics as discussed in the previous section, point to the use of feedback to control the output light accurately. The idea is to have a direct feedback loop from the optical power output to the electrical input signal. Prof. Vladimir Bulovic, from MIT Electrical Engineering Department, first suggested this idea.

3.1. Transparent OLED

The feasibility of this idea is suggested in his paper [2], where transparent OLEDs were demonstrated. Transparent OLEDs differ from common OLEDs in that they can be observed from both sides, i.e. light output can be observed from both the front and the back. Figure 7 shows an alphanumeric display made from transparent OLED pixels. Notice that the lettering in the background is observable even when the OLED pixels are on.

Figure 7. An Alphanumeric Display Made from Transparent OLEDs [2]

Since most flat panel display applications do not need the advantage of being able to view the display output from both sides, we have extra leverage: complete access to one side of the display output. The idea is to put a silicon photodetector at the back of each transparent OLED structure to directly sample the optical power output level and convert it to a useable electrical feedback signal to control the OLED driving current as shown in Figure 8.

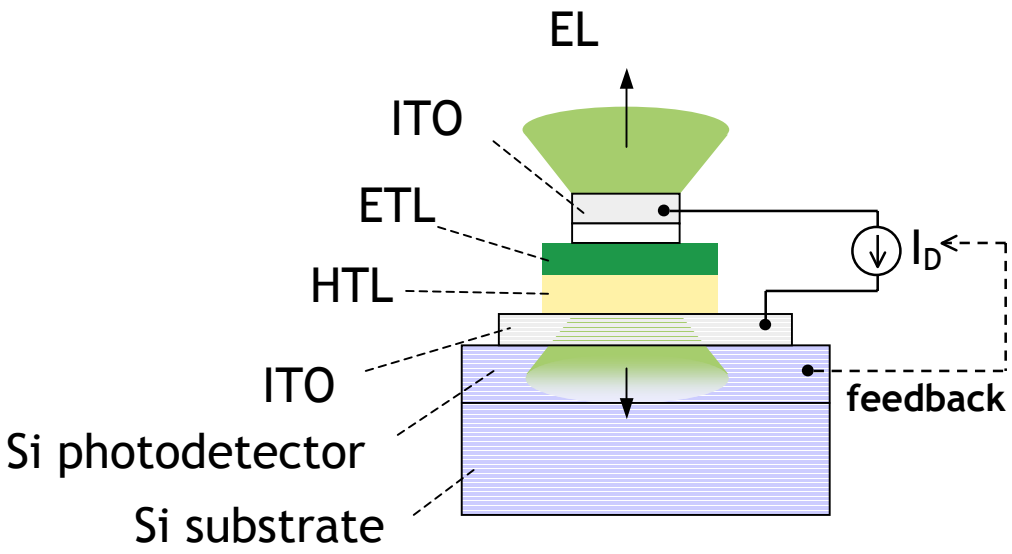


Figure 8. Transparent OLED on a Silicon Photodetector

3.2. Direct Optical Feedback

The electrical feedback signal, which will represent the light output intensity level, is then used to control the driving signal so that the output optical power consistently represents the input reference signal. Figure 9 shows the block diagram for this idea. The idea has the potential to succeed since the sensor can be designed to have a much more reliable and consistent characteristic compared to the OLED.

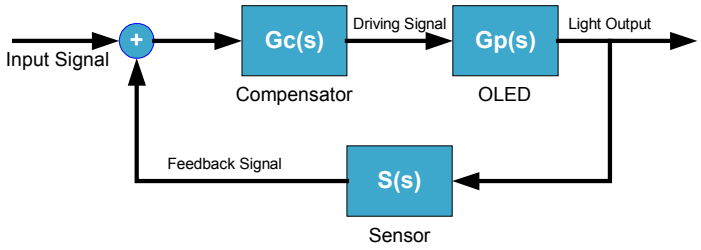


Figure 9. The Block Diagram of Main Feedback Idea

The goal of the thesis is to create a working 5x5 pixels OLED display, which maintains uniform grayscale reliability despite the varying characteristics of the individual pixels. The final demonstration system includes the 5x5 pixels OLED based displays together with the addressing, the feedback and the driving circuitry implemented using discrete components.

3.3. A Feedback Loop Shared by a Column of Pixels

There are several considerations to be made for the feedback loop implementation. Since the demonstration system is geared to building a model for the later integrated implementation, there are many more aspects to be considered. Ideally the discrete demonstration implementation should: use the simplest circuits possible, use as small number of devices as possible, be low power so that the power efficiency potential of OLED-based displays can be achieved and scalable to a much larger number of pixels.

The simplest implementation of the feedback loop of the display system will be to have a loop for every single pixel. However, this is expensive in term of the number of components, which translates to space and complexity if the design is used in an integrated version. Moreover, a continuous running feedback loop around every pixel will also tend to be expensive in terms of power since the feedback circuitry is also consuming power.

On the other hand, a display design based on a single feedback loop per pixel can be expanded easily to large number of pixels, as every pixel and its control loop is then simply an exact copy of another. Moreover, in the integrated implementation, the light sensor as shown in Figure 8 and 9 will be implemented using a simple silicon p-n junction. The close spatial proximity of the sensor to the feedback loop will make the sensing more accurate. As a result each pixel will have less error and more consistent output.

Another possibility is to have a small number of feedback loops, each reusable by a group of pixels using some addressing mechanism. This alternative has the potential of being lower in power consumption and in the number of devices.

However, with this scheme there are extra requirements on the feedback loop since each of the pixels only has access to the feedback loop for a limited of time within each cycle. In other word, the feedback loop must have a faster step response (larger bandwidth). Furthermore, the pixel design also has to include a relatively accurate sample and hold circuit so that it can reliably store a driving signal set by the shared feedback loop and maintain it through out a full cycle of refresh time. The basic schematic for this shared feedback loop is shown in Figure 10.

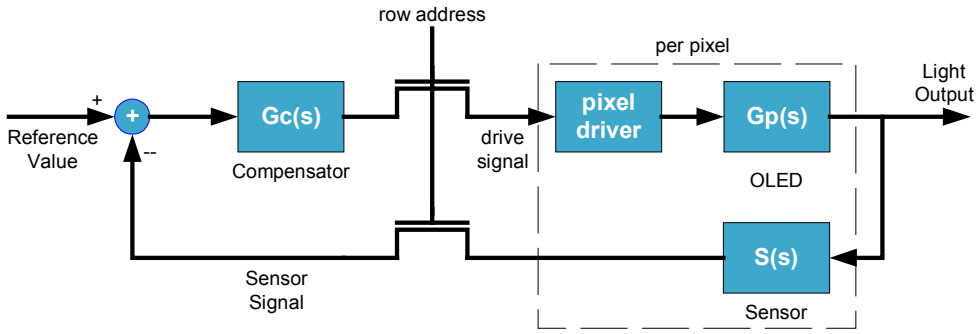


Figure 10. Simple Schematic for the Shared Feedback Loop Idea

In this thesis project, a single feedback loop shared by a single column of pixels is chosen as the method to drive the display because a single feedback loop per pixel turns out to be prohibitively expensive in terms of real estate and pixel complexity. Moreover, the driving circuitry in the feedback loop can use the conventional display driver circuitry since a loop per-column topology means that the display is refreshed in a row by row fashion similar to the active matrix topology in the commercially available LC based display. This also means that the same buffering and data format used in any active matrix display can be used to drive the proposed OLED based display. In the demonstration system, a single feedback loop for each column of 5 pixels is built, together with the sample and hold as well as the addressing circuitry.

Chapter 4

System Implementation

In this section, the system implementation is discussed. Since a lot of the work in this thesis is geared towards building the eventual integrated version of an Organic LED based display, a lot of the effort in the beginning is concentrated in the preliminary design of the pixel circuitry, feedback network and the array architecture for the integrated display. Therefore, in this section the preliminary design for the future integrated version is discussed first. After which, the 5x5 pixels demonstration system implementation is discussed.

4.1. Preliminary Integrated Design

The preliminary design was done in the early stage of the thesis as a starting point for the demonstration system. Moreover, the main purpose of the demonstration system is to learn as much as possible of the feasibility and the effectiveness of the direct optical feedback as will be in the integrated version. Thus, with the availability of this preliminary design, the demonstration system design can be made similar to the integrated design.

4.1.1. Feedback Network and Pixel Circuitry

Figure 11 shows the overall feedback loop for a single pixel as will be designed in the integrated implementation. The circuit in the dotted line is repeated for every single pixel. The feedback select path and the feedback signal path consists of the PMOS transistor m3 and the square silicon photo detector next to the OLED respectively. The capacitor and resistor are used to compensate the feedback loop to ensure stability and good error performance. However, depending on the architecture of the OPAMP used, the compensation network may be simpler or even not necessary as would be case if an output load compensated OPAMP is used.

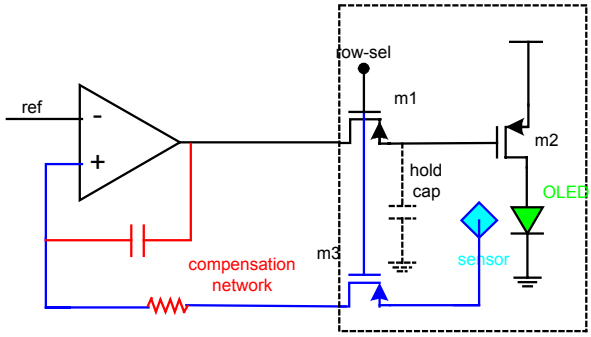


Figure 11. Integrated Single Pixel Feedback Loop Schematic Details

Note that the hold capacitor is drawn with dotted lines because in the integrated implementation, the gate capacitance of the driving transistor may be sufficient to act as the hold capacitance. This capacitance will hold the voltage at the gate of the driving transistor over a single refresh cycle.

4.1.2. Pixel Array

The pixel circuitry and the feedback loop shown in the previous section is then replicated and arranged to form the pixel array and its driving circuit. The simplified schematic of a 5x5 pixel array is shown in Figure 12. Note that the compensation network is not included in the figure but they are implicit in the OPAMP block.

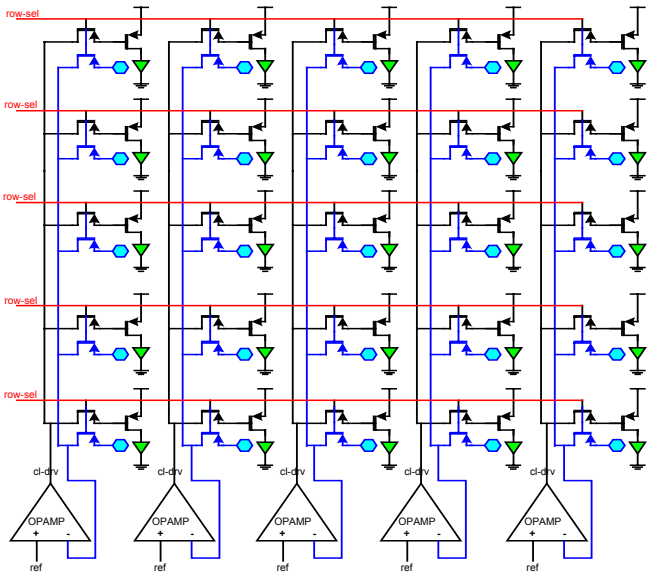


Figure 12. The Schematic of an Integrated 5x5 Pixels Array

4.2. 5x5 Pixels Demonstration System

Figure 13 shows the overall system block diagram for the demonstration system. The system can be generally divided into two large parts: analog and digital. The analog part is responsible mainly for the pixel circuitry, which includes the sample and hold (S/H), as well as the feedback loop and its compensation network. The digital part is responsible for the sensing (the CMOS camera in this case) and the control circuitry (implemented using Complex Programmable Logic Devices - CPLD).

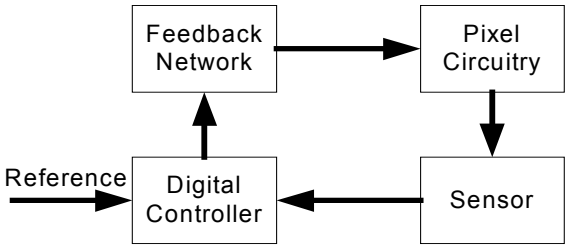


Figure 13. The Overall System Block Diagram

4.2.1. Feedback Network and Pixel Circuitry

In this demonstration system, the sensor and the feedback path is implemented digitally using a CMOS camera and a CPLD. The overall schematic for the demonstration system feedback loop with a single pixel is shown in Figure 14. The digital reference signal and the digital feedback signal are then converted to an analog signal before fed into the driving op-amp. The digital feedback implementation is chosen based on the simplicity and availability of the CMOS camera.

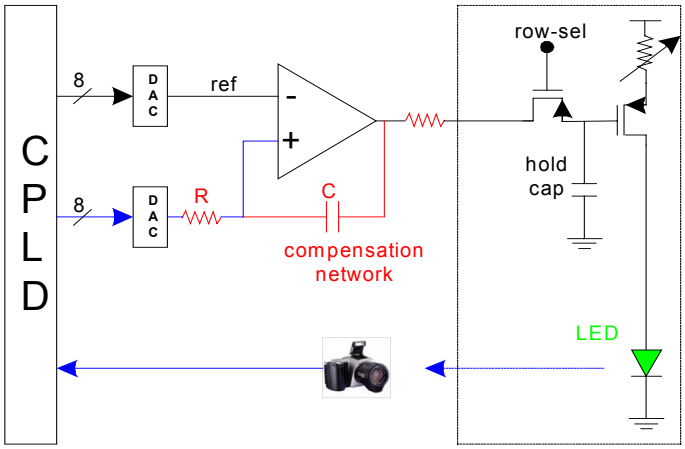


Figure 14. Demonstration System Feedback Loop Schematic

Note that when the project was completed, the 5x5 array of OLED pixels was not available. Therefore, common discrete LEDs (silicon based) are used instead since the behavior of the OLEDs can be emulated with these LEDs and a source resistance to the driving PMOS transistor. Moreover, the proof of feasibility of the optical feedback loop using the circuit topology shown in Figure 14, which is the final target of the thesis can still be obtained as shown later in Chapter 5.

4.2.1.1. Pixel Circuitry

The detailed version of the implemented pixel circuit, together with the component types and values are shown in Figure 15. The driving PMOS transistor is used as a current source to drive the OLED. The driving PMOS transistor is chosen based on its current carrying capability and operating voltage. The LED used required 5 to 30mA of current for nominal observable brightness and the operating voltage across it is 2 V at maximum brightness. The same type of PMOS transistor is then used for row-select switch since this switch performance is not critical in the feedback loop.

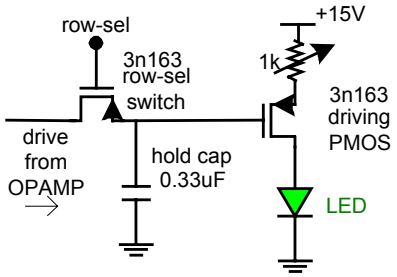


Figure 15. Pixel Implementation Schematic Details

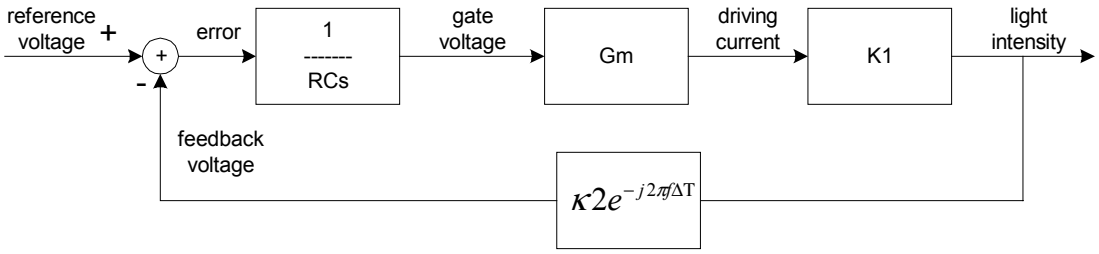
The 0.33uF capacitor value is chosen as a convenient value so that the capacitor is large enough to be able to hold the necessary gate voltage value so that the driving PMOS transistor can maintain a fixed current through the LED over a single refresh period.

The pixel circuit works as follow: When the *row-sel* signal goes low, the row-sel switch conducts, allowing the *drive* signal from the OPAMP to alter the voltage value at the hold capacitor and hence the voltage at the gate of the driving PMOS transistor. The gate voltage in turn affects the current through the LED. When the *row-sel* signal goes high, the row-sel switch stops conducting and the hold-capacitor holds the voltage at the gate of the driving PMOS transistor until the *row-sel* signal goes low again.

4.2.1.2. The Feedback Loop

The OPAMP driving signal is controlled according to the reference signal as shown in Figure 14. When the reference signal increases, the OPAMP output will decrease thereby setting the driving PMOS transistor current higher and the LED brighter. Similarly when the OPAMP reference signal decreases, the OPAMP output will increase, which sets the driving PMOS transistor current lower and the LED dimmer.

The block diagram of the feedback loop is shown in Figure 16. The camera, the CPLD and the DAC in the feedback path is modeled as a gain with a time delay. The gain term has a unit of voltage per candela/m^2. The time delay term is mainly caused by a number of frames buffering in the CMOS camera [16] and in the camera processing board [17]. These together with the frame buffering in the CPLD is modeled conservatively as three times a frame period:



$$\Delta T = 3 \times \frac{1}{60} = \frac{1}{20} = 50ms \cdot$$

Figure 16. Block Diagram of the Feedback Loop

The current to light conversion of the LED is also modeled as a linear gain term, with units: (candela/m^2)/A = candela/A.m^2. Note also that higher order poles due to the parasitic capacitances of the PMOS transistor, poles of the OPAMP as well as the hold capacitor are ignored since the overall system bandwidth is set to be much lower than these poles to ensure that the negative phase shift contribution from the time delay in the feedback path does not cause the feedback loop to be unstable.

4.2.1.3. Compensating the Feedback Loop

Instead of figuring out the value of K1 and K2 from the small signal analysis of the devices, a more practical approach is taken. The linear gain terms: Gm, K1 and K2 are lumped into a single gain term K. Note that as the devices degrade, the value of K will decrease. Since the loop is a single pole loop, reducing K will increase the stability of the loop.

Therefore, new devices are obtained (maximum K) and a single loop is built. Remembering that the loop transmission is: $L(s) = \frac{Ke^{-j2\pi f\Delta T}}{RCs}$, the value of RC is changed until a step response with an estimated phase margin of 60° is obtained. Note that ΔT=0.05s, therefore for a phase margin = 60° the value of the cross over frequency is set to be $f_c = \frac{[180-(90+60)]}{360\times\Delta T} = 1.67Hz$, which implies that $\omega_c=2\pi f_c=10.5$rad/s. As a reference, a step response of the closed loop shown in Figure 17 is obtained using MATLAB. Note that the value of G is set to be 10.5 so that the cross over frequency of the loop is 10.5 rad/s.

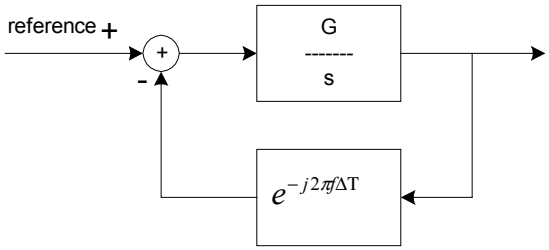
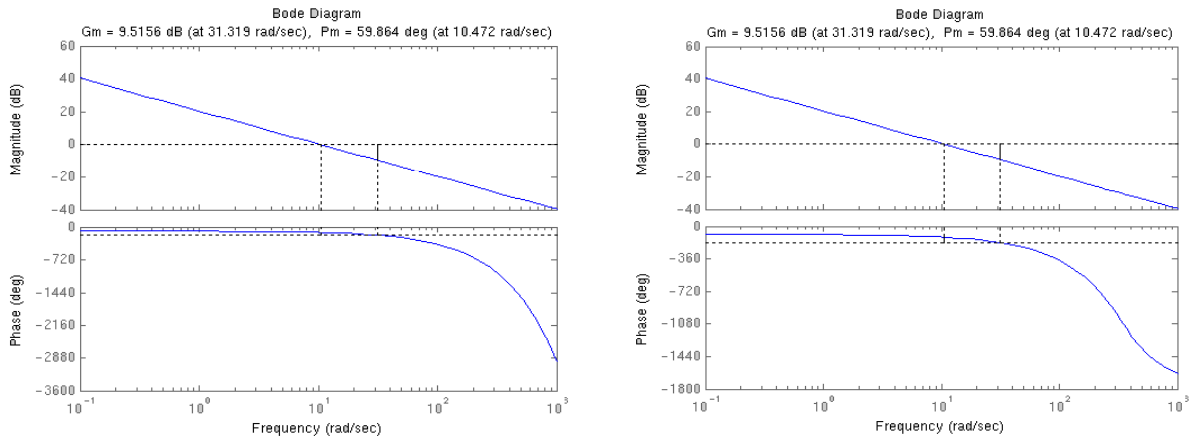


Figure 17. Reference Loop Used to Estimate Step Response

Since MATLAB does not handle pure time delay in the loop for a closed loop response, the time delay is approximated using PADE formulae [18] to get a rational function whose magnitude and phase behavior accurately emulate the pure delay at low frequency. Figures 18(a) and (b) compare the Bode plots of the loop transmission of system with pure time delay and with the PADE approximation of order 10.



(a) (b)

Figure 18. (a) Loop Transmission Bode Plot with Pure Time Delay
(b) Loop Transmission Bode Plot with PADE Approximation to the Time Delay

Note that both Figure 18(a) and (b) shows a phase margin of 60.0^{o} and a gain margin of 9.5dB. Therefore, the approximation is accurate especially at low frequency below the cross over frequency. Figure 19 shows the step response of the closed loop feedback system using the PADE approximation to the time delay.

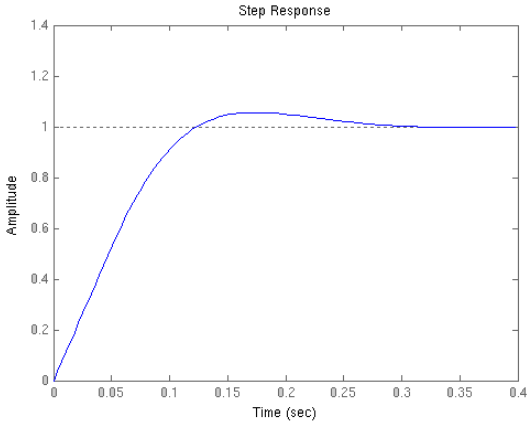


Figure 19. Step Response of the Closed Loop System with ϕm=60^{o}

Using the step response in Figure 19 as a reference for the closed loop small signal step response, the value of the RC product in the system as shown in Figure 14 and 16 is adjusted. The value of the RC product is varied until the time constant and percentage overshoot of the small signal system is close to that of Figure 19. In Figure 19, the time from the beginning of the step to the first peak is approximately 150ms and the overshoot is approximately 10%. The final

value of R and C obtained are: R=820KΩ and C=0.33uF. The small signal step response behavior with these values of R and C is shown in Figure 20, with rise time from 0 to first peak of approximately 144ms and overshoot of 16%.

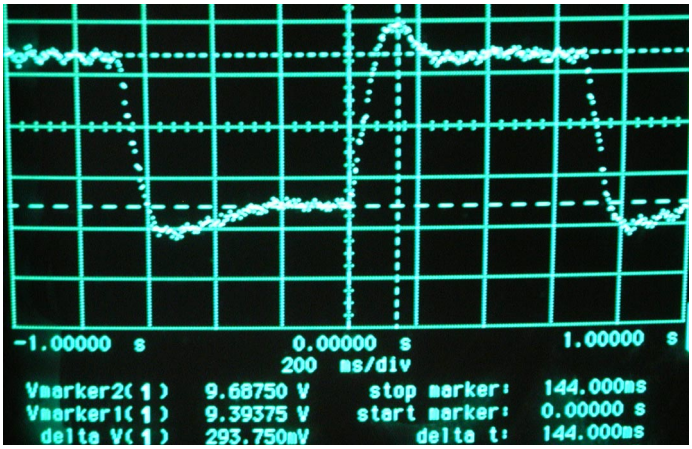


Figure 20. Step Response of the Actual Closed Loop System with R=820KΩ and C=0.33uF at Maximum Brightness Operating Point

By setting the value of the RC product as such, the bandwidth of the system is set to be approximately 1.7Hz as discussed earlier. Note that the maximum brightness operating point is chosen when adjusting the resistor and the capacitor values. This is necessary since the value of Gm (transconductance of the driving PMOS transistor) changes with the operating current, and hence the operating brightness. The value of Gm decreases with decreasing current. Therefore, the small signal step response for compensation is obtained at the worst-case condition for stability, i.e. when Gm is maximum, which occurs at maximum brightness. Maximum Gm corresponds to the worst-case condition for stability since a higher Gm leads to a higher loop gain, which in turn leads to a higher cross over frequency (Figure 18(a)).

When a small signal step is applied at minimum brightness, the response is expected to have less peaking, which corresponds to higher stability. This is indeed the case as seen in Figure 21. The higher stability also corresponds to a lower bandwidth and hence a higher rise time, which is close to 200ms compared to approximately 100ms at maximum brightness operating point.

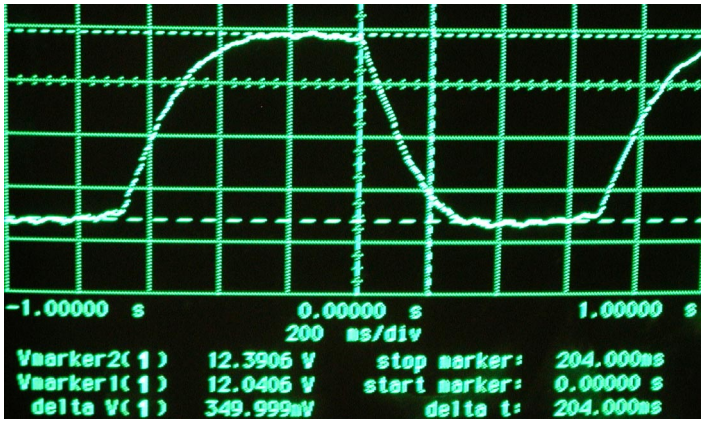


Figure 21. Step Response of the Actual Closed Loop System with R=820KΩ and C=0.33uF at Minimum Brightness Operating Point

After obtaining the small signal step responses at different operating points, a large signal step response is obtained to ensure a good behavior at large transients. The step response, giving a 4.28V step at the gate voltage, is shown in Figure 22. The rise time is approximately 150ms, which is an average of the rise times at maximum brightness and minimum brightness. This is reasonable since the step moves the system from the minimum brightness operating point to the maximum brightness operating point.

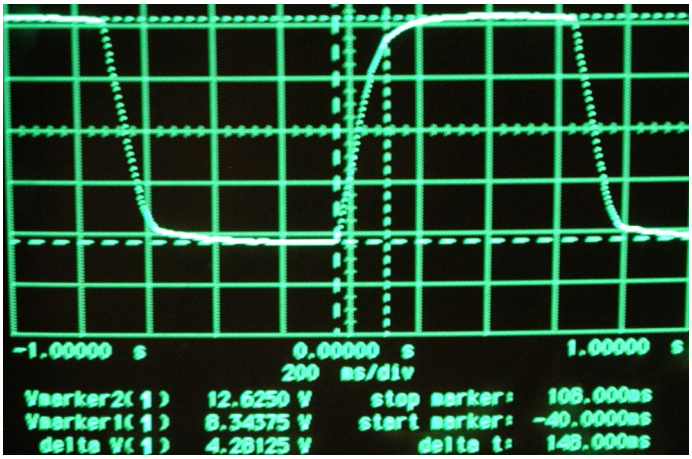


Figure 22. Large Signal Step Response

Note that all the traces are obtained using a digital scope, with 2048 samples averaging. The small ripples noticeable at the DC settling point of the small signal step responses is attributed to noise and are averaged out as more samples are used.

Notice that in Figure 14, there is a resistor in between the output of the OPAMP and the row-sel switch. The purpose of the resistor is to act as a low pass filter in combination with the hold capacitor, to filter out the small step changes in the output of the OPAMP as it is switched

from one pixel in a certain row to another pixel in another row. The value of this resistor is set to be small to ensure that it does not affect the dynamics of the close loop system significantly. In this case, a value of 1.30KΩ is used (which causes its RC product with the hold capacitor to be 3 orders of magnitude lower than the main compensation network).

In this system, the slow response of the loop is mainly caused by the delay in the feedback path, which is mainly due to the frame buffering in the camera and its processing board. This delay will not be present in the integrated implementation and hence the loop response can be made much faster.

However, one important observation from the small signal responses is that in the integrated version, the feedback loop has to be compensated such that it is fast enough to meet the refresh period requirement at the lowest operating Gm and it is slow enough such that the loop is stable at the highest operating Gm. Moreover, the feedback loop has to satisfy these requirements as the OLED degrades, which lowers the open loop gain of the system.

Finally Figure 14 is repeated here in Figure 23 with the type and values of all the components used in the demonstration system included.

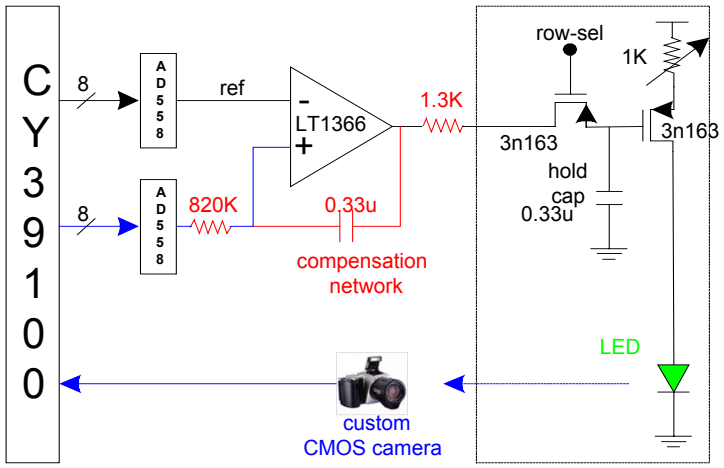


Figure 23. Demonstration System Feedback Loop Schematic with Component Types and Values

4.2.2. Pixels Array

The pixel circuitry and the feedback loop discussed in the previous section is then replicated and arranged to form the 5x5 pixel array and its driving circuit. The simplified schematic of a 5x5 pixel array is shown in Figure 24. Note that the compensation network is not included in the figure but they are implicit with the OPAMP block.

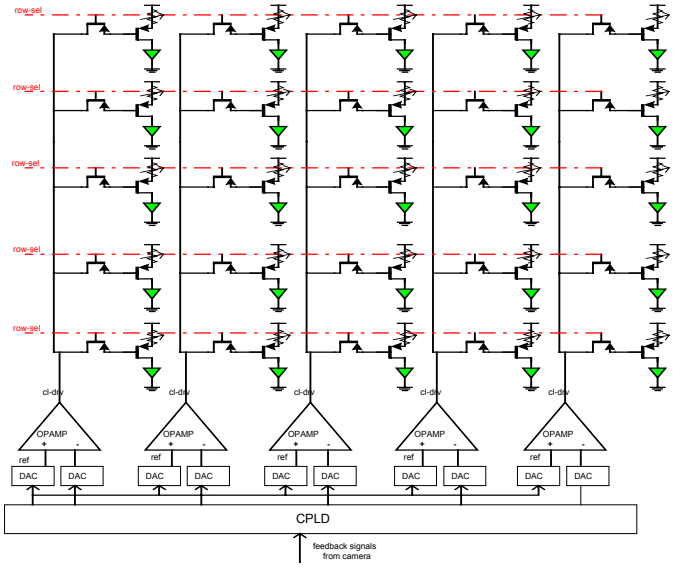


Figure 24. The Simplified Schematic of the 5x5 Pixels Array with the Driving Circuit

The feedback paths in Figure 24 are represented by the output from the camera to the CPLD. Notice that a single OPAMP is shared among 5 pixels in a column. Since, at any point in time, only one *row-sel* signal is low, the pixels are updated one row at a time.

The *row-sel* signal and the mapping of the camera output to each of the individual pixel are provided by the CPLD. The CPLD simply updates one row at a time, going from row 1 to 5 and going back to row 1. When it updates one row, it waits until the feedback loop settles to its final value before going to the next row.

4.2.3. Digital Interface and Control

After discussing the mostly analog feedback loop and pixel array of the system, the digital camera and the digital control part of the system will be explained next. The digital control mainly involved a large state machine implemented using a CPLD using signals out from the camera processing board.

4.2.3.1. The Digital Camera

The digital camera used is based on a CMOS imager chip developed by Decker and Sodini [16]. The wide dynamic range capability of the pixels has been turned off to ensure a linear response between light intensity and voltage output.

The imager chip processing board used is based on a decoder board developed by Fife and Sodini [17]. Again most of the features in the board are turned off to ensure linear response between light intensity and the voltage output. From here onward, the camera and its processing board will be referred together as the camera.

The choice of the digital camera is solely based on availability at the time when this thesis was developed. Any other type of camera with an NTSC output in combination with an NTSC-to-digital encoder chip or any camera with a digital output could have been used. The only important criteria for the digital camera is the uniform characteristics of its pixels across the whole array of the imager chip so that the light intensity input to output voltage function does not depend on the position of the pixel in the array.

From the system point of view, the camera can be viewed as a black box, giving out the voltage level of each of its pixels, which corresponds to the light intensity of an image in its field of view. The signals out of the camera used in this demonstration system are shown in Figure 25.

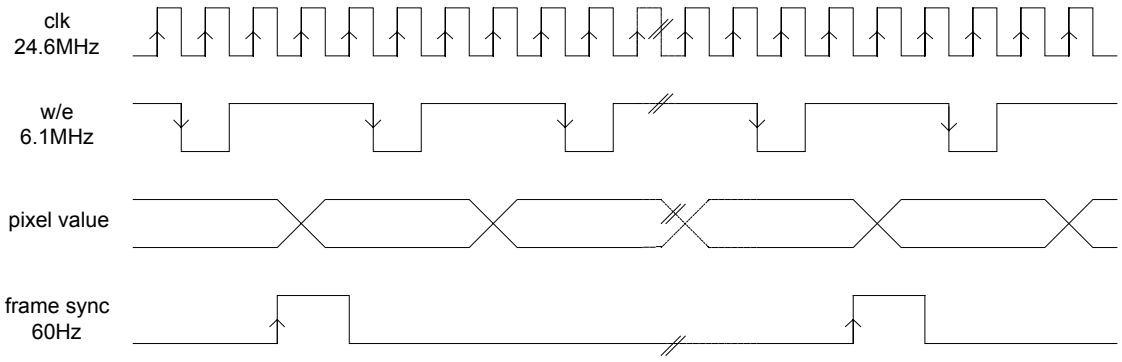


Figure 25. Output Signals from Camera

The base clock from the camera, which is used as the base clock for the whole system, has a frequency of 24.6MHz. The falling edge of the write enable signal (*W/E*) indicates that a pixel value is ready to be read. The *W/E* signal runs at 6.1MHz. And the rising edge of the frame sync signal, as the name indicates, signifies the beginning of a frame. The frame sync signal has the standard NTSC non-interlaced refresh rate of 60Hz. Including the time period required for vertical and horizontal blanking signal, the whole frame period contains 256x256 usable pixel values.

4.2.3.2. Digital Control

The overall functional block diagram of the digital control block of Figure 13 is shown in Figure 26. The digital control block takes in the signals from the digital camera and outputs the signals required by the array as shown in Figure 24. These signals include *row-sel* signal, the reference-value signal as well as the corresponding pixel values required by each of the column drivers (in the feedback path in Figure 24).

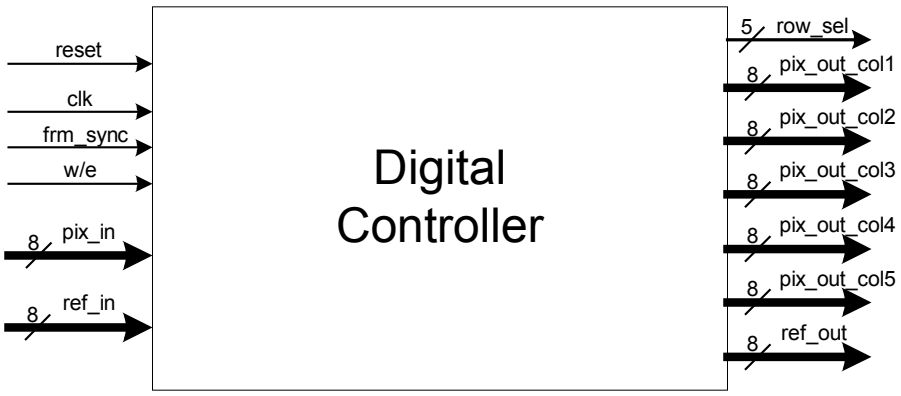


Figure 26. Overall Functional Block Diagram of the Digital Controller

4.2.3.2.1. Overview of the Digital Controller

The details of the digital controller including its sub-modules are shown in Figure 27 (on the next page). The controller's state machine manages all the sub-modules to give the desired behavior. The controller first undergoes a calibration procedure to map the camera image pixels to the position of the LED pixels. The address of the camera pixels corresponding to the position of the LED pixels is then saved in the memories. During feedback, the controller simply uses the saved addresses to observe the brightness of the LED pixels and feeds this information to the column driver OPAMP.

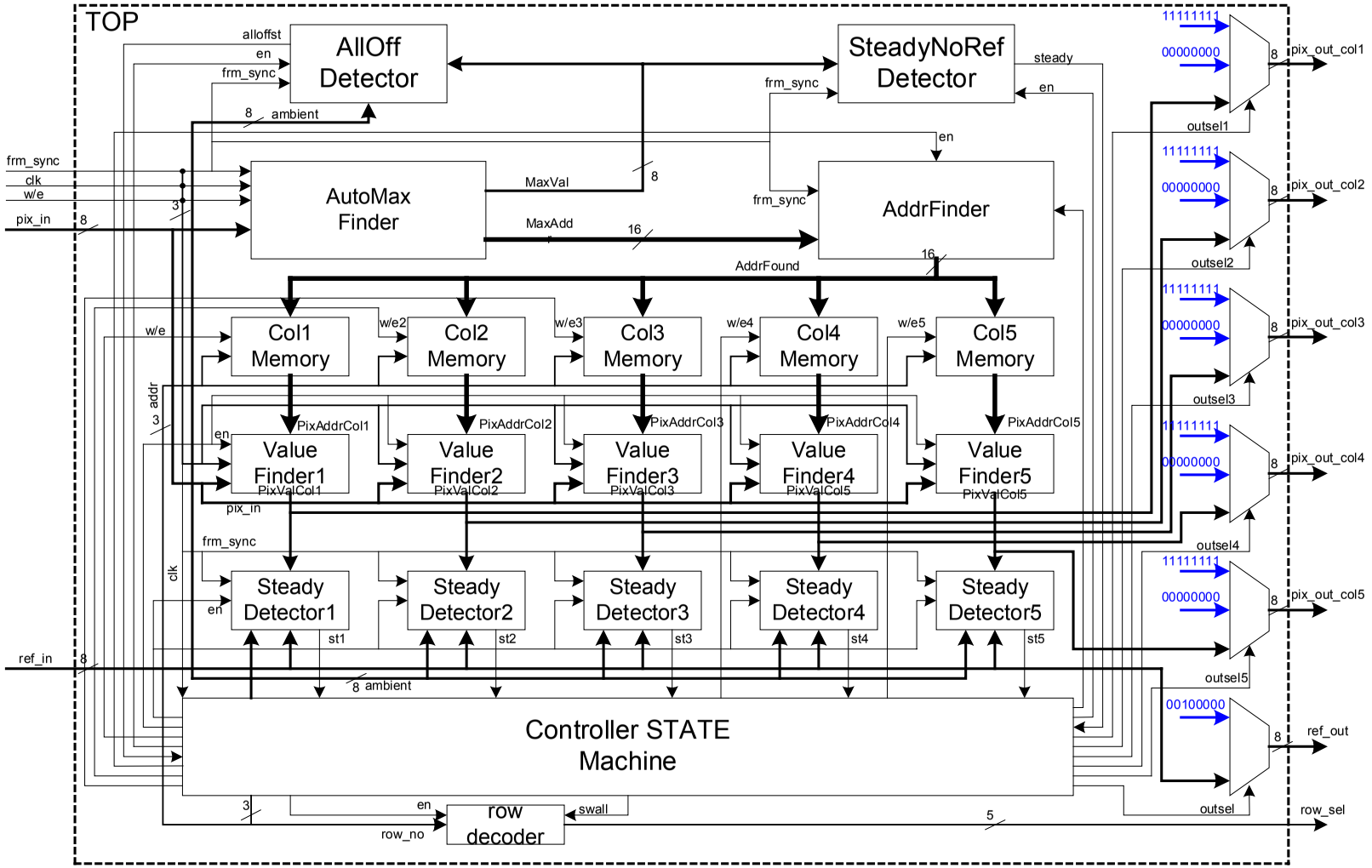


Figure 27. Detailed Functional Schematic of the Digital Controller

The controller uses the various modules to facilitate its function. For example, the AutoMaxFinder module is used to find the brightest spot in the camera field of view, which is useful during calibration. The other modules are the state-machine, the memories, and the various detectors whose functions are explained in detail next.

4.2.3.2.2. The State Machine and The Modules

The state machine, as the main component of the digital controller, will be discussed first. The state diagram of the state machine is shown in Figure 28. The VHDL code that implements the state machine and the modules are included in Appendix A. The state machine has in total 8 states: Idle, GetAmbient, SetPix, FindAddr, StoreAddr, SwAllOff, Feedback and FeedReset.

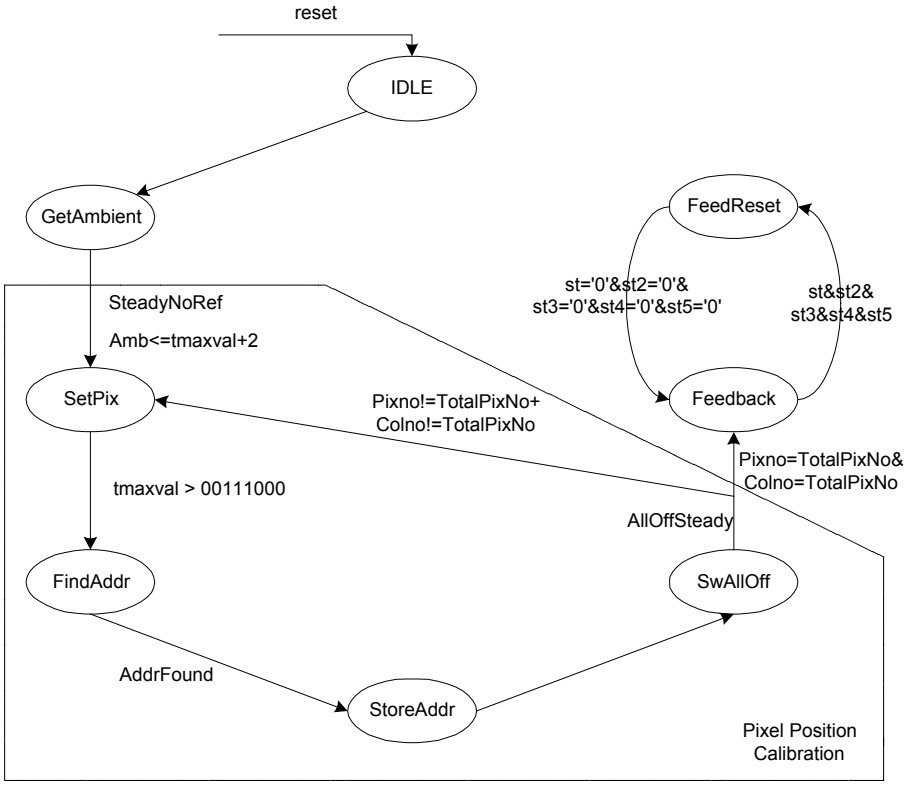


Figure 28. State Diagram for the Digital Controller State Machine

The condition adjacent to an arrow indicates when the transition between any two states occurs. For example, when *AddrFound* is set to '1', the state transition from FindAddr to StoreAddr at the next rising edge of the clock. The exception is the Idle state; whenever *reset* is set high the Idle state is entered next, regardless of the current state of the state machine.

When there is no condition listed next to an arrow, the transition occurs at the next rising edge of the clock. For example, Idle state always transitions to the GetAmbient state at the next clock edge. For the following discussion on the state machine and the various modules, refer to Figure 27 and Figure 28.

At the idle state, all signals are set to its initial values. The output pixel values for all the columns are all set to "11111111", the *ref_out* value is set to "00100000" and all the *row_sel* bits are set high. By setting this condition all the LEDs are switched off, since the driver OPAMP perceived that the feedback signals are greater than the reference signal.

At the next clock edge, the GetAmbient state is entered. This state is used to obtain the maximum ambient light that is present in the current field of view of the camera. This is to ensure that the digital controller knows the minimum brightness that it can differentiate from the system's surrounding. The controller obtained this ambient level by keeping all the LEDs off and getting the output value from the AutoMaxFinder module, whose function is to return the maximum pixel value in the camera's viewing field and that particular pixel address. The ambient level is deemed to be found when the output value from the AutoMaxFinder is steady for at least 16 consecutive clock edges. The SteadyNoRefDetector module detects this steady state of the ambient level. The module name indicates that the detector simply ensures that a particular value of the ambient level is repeated enough times before deciding that a particular value is in fact the steady ambient level. This is in contrast to the SteadyDetector module that compares the value it receives with a particular reference value.

When the SteadyNoRefDetector indicates that it finishes detecting the ambient level by setting the *SteadyNoRef* signal high, a variable is set to a value slightly above the ambient value obtained and this sets the minimum brightness that the controller will try to differentiate from its' surrounding. After the minimum brightness is set, the SetPix state is entered.

This state is used to set a single LED brightness, so that the controller can find which of the camera pixels corresponds to that particular LED. In this state, only one of the bits in the *row-sel* signal is set high and only one of the pixel values output is set to "00000000" (while maintaining the other pixel value outputs at "11111111"). Since the *ref_out* value is maintained at "00100000", one of the LEDs will be set bright since the OPAMP driver perceived that its feedback signal is lower than the reference value. On the other hand, all the other LED will remain dark since either its *row-sel* bit is not set high, maintaining its brightness from the previous state; or its pixel value output is set at "11111111".

When the state machine detects that the value returned by the AutoMaxFinder is greater than “00111000”, indicating that one of the LEDs has been set bright enough, the state machine enters the FindAddr state. In this state, the AddrFinder module is enabled. The AddrFinder is used to ensure that the address of the camera’s pixel with maximum brightness from the AutoMaxFinder module is steady. This is to prevent the controller from getting the wrong address during transient. The AddrFinder ensures the steadiness by making sure that the maximum pixel address is not changing for a number of clock cycles. When the AddrFinder module finds the address steady, it sets the variable *AddrFound* high.

Once the *AddrFound* variable is set high, the state machine enters the StoreAddr state. In the StoreAddr state, the address from the AddrFinder is stored in the “column memory”. The appropriate “column memory” is write enabled by setting the *W/E(x)* signal high. The value is then stored at memory address corresponding to the current row number, by putting the row number into the address bus. Since the memory is written within a single clock cycle, the state machine transitions to SwAllOff on the next clock edge.

In the SwAllOff state, all the LEDs are turned off in a similar way as they are turned off in the Idle state. Then the AllOffDetector is enabled. The AllOffDetector ensures that the output of the AutoMaxFinder is less than or equal to the value stored in the *ambient* variable for a sufficient number of clock cycles. When this condition occurs, it means that all the LEDs are turned completely off. Then the AllOffDetector module sets the *AllOffSteady* variable high.

When *AllOffSteady* is high, the state machine runs some test to check whether it has finished calibrating the position of all the 25 LEDs. The state machine does this by checking whether the current row number and the current column number are equal to the total row number and the total column number respectively. If not all the 25 LEDs are calibrated, the state machine either increments the current row number, or increments the current column number and resets the current row number to 1. The state machine then transitions back to the SetPix state and the whole process repeats until all 25 LEDs are calibrated. When all 25 LEDs are calibrated, the state machine transition to the Feedback state.

In the Feedback state, the state machine uses the addresses stored in the “column memory” to output the appropriate values to the OPAMP column driver. In order to be able to do this, the state machine enables the ValueFinder module. This module returns the value of the pixel referred by the address supplied to the module.

During the transition from the SwAllOff state to the Feedback state, the *row_no* variable is reset to "001". Therefore, the first time the Feedback state is entered, the pixel values corresponding to the 5 LEDs in the first row is output from the digital controller since the *row_no* variable is also used as the address to read off the "column memory".

During the Feedback state, the SteadyDetector modules are also enabled. The function of this module is to ensure that the feedback loop settles before the state machine switches the loop to the next row. It does this by ensuring that the output values from the ValueFinder modules are steady for a certain number of clock cycles. When all the SteadyDetector modules reported that the pixel values they are observing are steady, the state machine transition to the FeedReset state.

In this state, all the SteadyDetector modules are reset and the Decoder module is disabled. When the Decoder module is disabled, it sets the *row_sel* bits all to low causing none of the row to be connected to the feedback loop. Once all the SteadyDetector modules are reset, indicated by the variables *st1* to *st5* set to '0', the state machine transitions back to the Feedback state.

During this transition, the *row_no* variable is incremented so that the next row is connected to the feedback loops. The whole process repeats from row to row. When row 5 is reached, the *row_no* variable is reset back to 1. The process continues forever unless *reset* is set high, in which case the state jumps back to Idle to begin the whole calibration process anew.

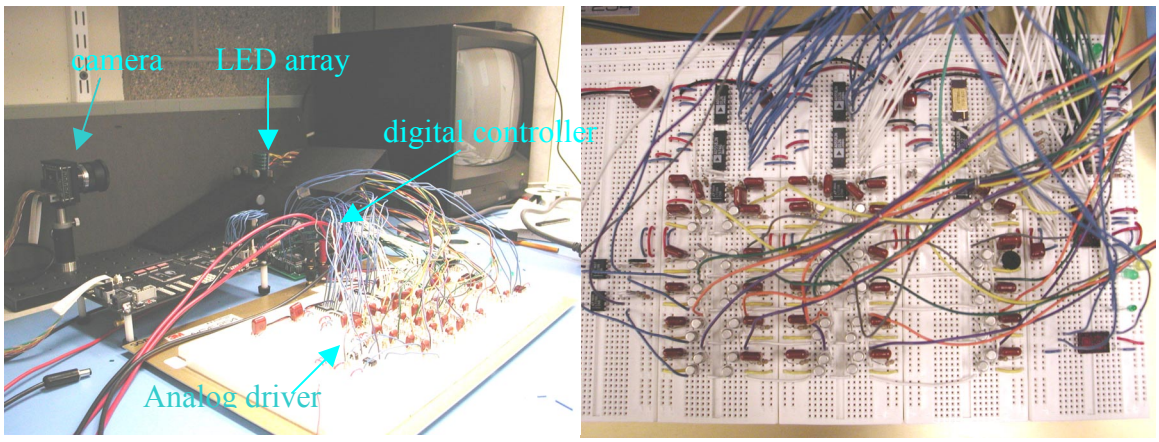
Chapter 5

Results

After discussing the details of the implementation, the results of the system implementation are presented in this section. Two versions of the system are built. They are identical in terms of components and function. The only difference is that one is built on a proto-board while the other is built on a printed circuit board (PCB). Section 5.1 will show some pictures of the proto-board implementation. Section 5.2 shows the printed circuit board used in the final demonstration system. Section 5.3 discusses some of the images, obtained using the demonstration system, proving the feedback concept as well as the feasibility and the effectiveness of the direct optical feedback for controlling the Organic LED based display.

5.1. Proto Board Prototype

Figure 29(a) and (b) shows the proto board implementation of the system. The system is first built on a proto board to ensure fast turn around and ease of design modification. With such advantages, design and implementation can go hand-in-hand, allowing quick results and a more practical approach to the design such as one used to compensate the feedback loop (Chapter 4, section 4.2.1.3)



(a) (b)

Figure 29. (a) The Proto Board Implementation of the System
(b) The Details of the Analog Driver

5.2. Printed Circuit Board

The printed circuit board is designed using PCAD. The schematic and the layout plot built in PCAD for the circuit board is included in Appendix B. Figure 30 shows the final circuit board used in the demonstration system.

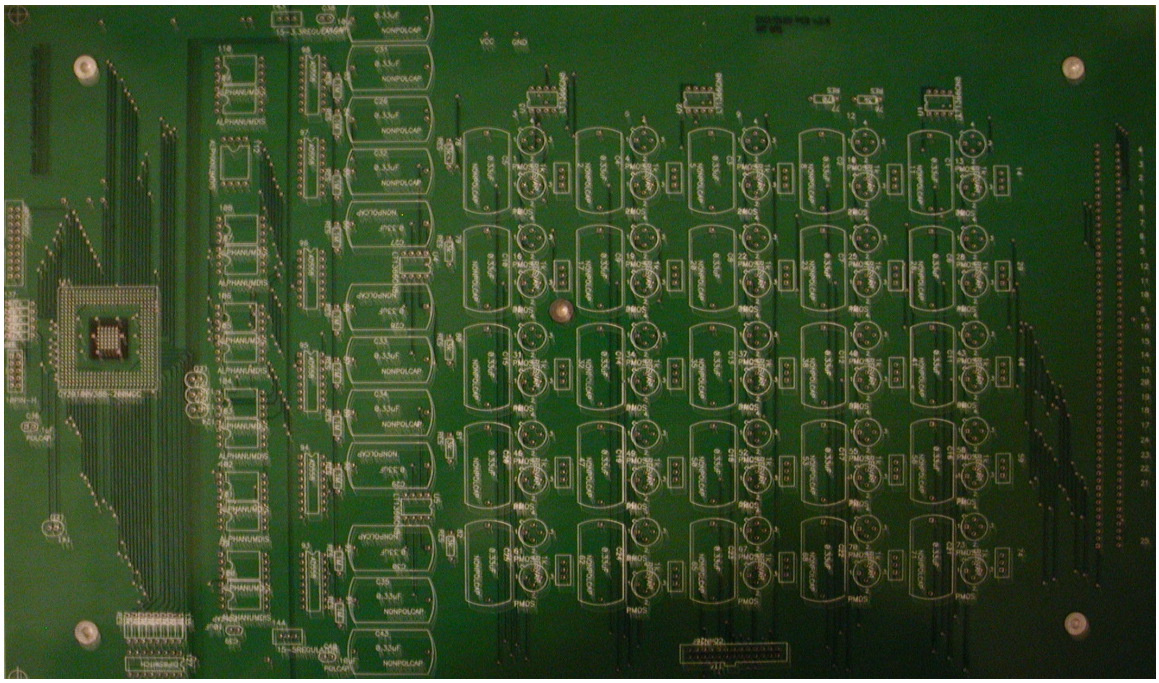
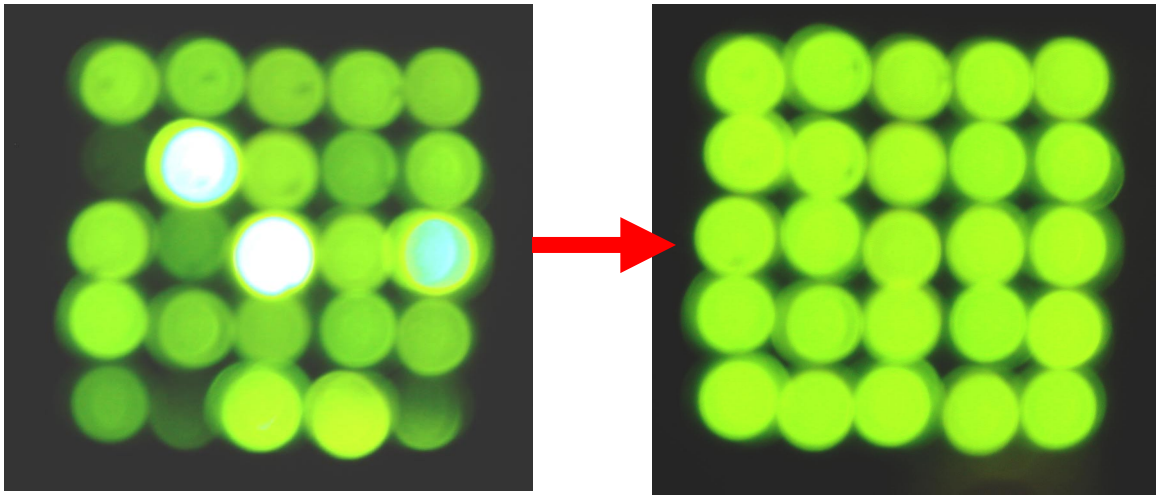


Figure 30. Final Demonstration System Printed Circuit Board

5.3. Display Uniformity with Feedback

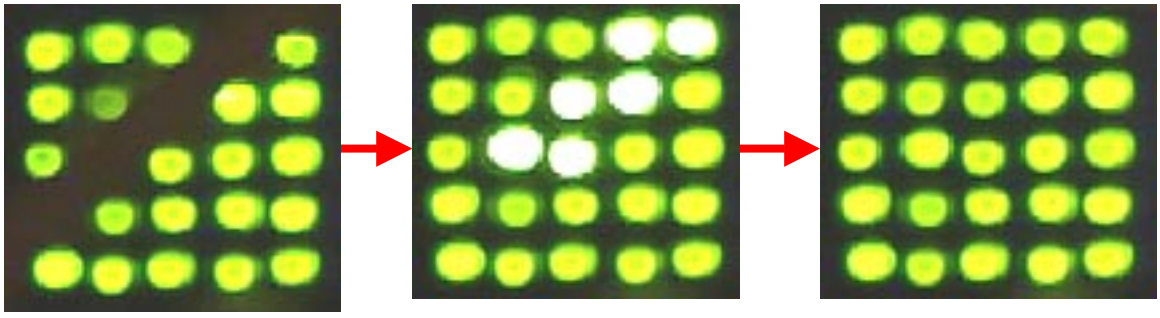
Figure 31(a) and (b) contrasts two images, showing the feedback at work. Figure 31(a) shows a pixel array with pixels of varying brightness. A few LEDs in the array are purposely made brighter or dimmer (by changing the series pot value in the pixel – Figure 14) to emulate irregularity in the Organic LED characteristics. Figure 31(b) shows the same pixel array showing a uniform brightness after the feedback system is applied to the array.



(a) Before (b) After

Figure 31. (a) Pixel Array Showing Non Uniformity in Brightness;
(b) The Same Pixel Array Showing Uniform Brightness After Feedback Correction

Figure 32 shows another result pointing to the effectiveness and feasibility of the feedback system. In Figure 32(a), an object was placed in between the camera and the LED array covering a number of the LED array pixels. The camera, unable to receive signals for the covered pixels, output low values to the digital controller. The digital controller in turn passed on this information to the OPAMP column driver. Receiving these low values, the OPAMP drives the LED harder to make it brighter. This effect is apparent in Figure 32(b), showing the array immediately after the object covering the pixel was removed. Once the whole system is allowed to settle, the pixels again shows uniform output across the whole 5x5 array.



(a) (b) (c)

Figure 32. (a) An Object Placed in Between Camera and LEDs Array Covering Some Pixels;
(b) The Same Array Immediately After Object is Removed;
(c) The Same Array after the Feedback Correction

Chapter 6

Conclusion

This thesis has described the design and discrete implementation of a feedback system to prove the effectiveness and feasibility of direct optical feedback for an organic LED based display. This section summarizes the project and presents suggestions for future work.

6.1. Summary

The Feedback method using an optical sensor proves to be an effective and feasible method to maintain grayscale reliability of an Organic LED based display in the face of the individual OLED electrical and power output characteristics variation. Many techniques have been developed previously to ensure a predictable current level of each individual pixel in a display array. However, none of the techniques takes into account the power output degradation and the I-V characteristics variation of the OLED device itself over time, temperature and process parameters. The objective of this thesis has been to demonstrate the effectiveness and the feasibility of direct optical feedback to overcome the limitations of the OLED devices as a candidate pixel device for high quality displays. This objective has been met through the design and discrete implementation of a demonstration feedback system capable of maintaining a uniform output pixel array in the face of emulated characteristics variation in the individual OLEDs. Besides achieving the main goal of the thesis, the project also provides results that can be applied to other systems requiring time-shared feedback loops for a number of output devices or actuators.

6.2. Future Work

The next immediate step after this project will be to build an integrated version of the system for a larger array of pixels with a standard display frame rate of 60 Hz. From this project, some of the further work that needs to be done in realizing the integrated version can be pointed out.

As mentioned before in Chapter 4 section 4.1.1, one of them will be building the OPAMP column driver. It has to be compensated fast enough such that the display meets the 60 Hz refresh rate requirement even at the lowest brightness level, where the loop gain is low. Yet the OPAMP also has to be compensated slow enough to be stable for the highest brightness level. Moreover, the two requirements have to be maintained as the OLED pixels degrade over time.

For a 1024x768 pixel display (XGA standard), a column of pixels will have 768 pixels. This means out off the 1/60 s for a whole frame, each pixel will have approximately 22us. Therefore the settling time at the lowest brightness level has to be set less than 22us. Assuming that the loop gain between the lowest and the highest brightness level differ by a factor of 10 and that the loop is maintained as a dominant pole compensated loop, the settling time at the highest brightness level will need to be 2.2us or less. This however, should not be a problem for the current 1.5u or lower CMOS technology, which can be used to make an OPAMP with a bandwidth larger than 5MHz. Moreover, there will be no pure delay term in the feedback path since there is no frame buffering involved such as in the discrete implementation described in this thesis.

Another part that needs to be considered in the integrated version will be the voltage levels required to drive the OLED devices. These devices required a large voltage drop across them ~ 9-12V. In the discrete implementation, there is no problem since the discrete PMOS transistor devices can withstand up to 30V across them. However, in the integrated version, if 1.5u or lower CMOS technology is used, the maximum voltage across typical devices will be on the order of 5V. Therefore, some kind of voltage level conversion and sharing between a number of devices need to be designed. A possible architecture for this voltage level conversion is to use charge pump.

Another important element that needs to be considered in the integrated version will be growing the organic materials on top of silicon substrates. In the integrated version, the optical sensor will be a P-N junction built on the silicon substrate. This sensor needs to be placed right behind the transparent OLED as was shown in Figure 8. The accuracy and the yield of such a process needs to be investigated further since it may have an impact on the possible size of the pixel and the necessary optical isolation distance between pixels placed in an array.

Once the integrated version is built, the next step will be to take the design and build full-fledged 15”-21” display. The circuitry for this large-scale display can be made using TFT technology such as those currently used to build the Liquid Crystal Display.

References

[1] Kaneko, E., "Liquid Crystal TV Displays: Principles and Applications of Liquid Crystal Displays", *KTK Scientific*, *Nikkei Electronics, Flat Panel Displays 1993* (Japanese Publication).

[2] V. Bulovic, P.E. Burrows, and S. R. Forrest, "Molecular Organic Light-Emitting Devices", *Semiconductors and Semimetals*, Vol. 64, 2000.

[3] J. Kido, and Y. Iizumi, *Appl. Phys. Lett*. **73**, 2721, 1998.

[4] M. A. Baldo, V. G. Kozlov, P.E. Burrows, M.E. Thompson, and S. R. Forrest, V.S. Ban, B. Koene, and M. E. Thompson, *Appl. Phys. Lett.* **76**, 4, 1997.

[5] H. F. M. Schoo, R. C. J. E. Demandt, J.J.M. Vleggaar, C.T. H. Liendenbaum, P. van de Weijer, and Y. Croonen, *Materials Research Society Meeting Spring 1998,* San Francisco, California, paper G1.1., 1998.

[6] G. Gu, P.E. Burrows, S. Venkatesh, and S. R. Forrest, and M. E. Thompson, *Appl. Phys. Lett.*, **68**, 2606, 1996.

[7] C. W. Tang, *Society for Information Display, International Symposium, Digest of Technical Papers* **27**, 181, 1996.

[8] C. W. Tang, and S.A. VanSlyke, *Appl. Phys. Lett.* **51**, 913, 1987.

[9] A. Miyaji, M. Yamaguchi, A. Toda, H. Mada, and S. Kobayashi, "Control and Elimination of Disclinations in Twisted Neumatic Liquid Crystal Displays", IEEE Trans. E. D. Vol. ED-24, No.7, 1977, pg. 811.

[10] P. E. Burrows, V. Bulovic, S.R. Forrest, L.S. Sapochak, D. M. McCarty, and M.E. Thomson, "Reliability and degradation of organic light emitting devices"*, Appl. Phys. Lett.* **65**, 1994.

[11] Bulovic, V. (1998), "The Role of Excitons and Interfaces in Molecular Organic Devices"*, Princeton University Ph.D. Thesis*, 1998.

[12] R. M. A. Dawson, Z. Shen, D.A. Furst, S. Connor, J. Hsu, M. G. Kane, R. G. Stewart, A. Ipri, C. N. King, P. J. Green, R.T. Flegal, S. Pearson, W.A. Barrow, E. Dickey, K. Ping, S. Robinson, C.W. Tang, S. Van Slyke, F. Chen, J. Shi, M. H. Lu, and J. C. Sturm, "The Impact of the Transient Response of Organic Light Emitting Diodes on the Design of Active Matrix OLED Displays"*, IEEE,* 1998.

[13] Pribat D, Plais F, "Matrix Addressing For Organic Electroluminescence Displays", *Web of Science,* 2001.

[14] R. M. A. Dawson, Z. Shen, D.A. Furst, S. Connor, J. Hsu, M. G. Kane, R. G. Stewart, A. Ipri, C. N. King, P. J. Green, R.T. Flegal, S. Pearson, W.A. Barrow, E. Dickey, K. Ping, S. Robinson, C.W. Tang, S. Van Slyke, F. Chen, J. Shi, M. H. Lu, and J. C. Sturm, "Active Matrix Organic Light Emitting Diode Pixel Design using Polysilicon Thin Film Transistor", *IEEE,* 1998.

[15] Yi He, Reiji Hattori, and Jerzy Kanicki, Current-Source a-Si:H Thin Film Transistor Circuit for Active-Matrix Organic Light Emitting Displays, *IEEE Electron Device Letters, Vol. 21, No. 12*, December 2000.

[16] Steven John Decker, "A Wide Dynamic Range CMOS Imager With Parallel On-Chip Analog-to-Digital Conversion", *Massachusetts Institute of Technology Ph.D. thesis,* September 1997.

[17] Keith G. Fife, "A Stereo Vision System with Automatic Brightness Adaptation", *Massachusetts Institute of Technology MEng thesis,* May 1999.

[18] Golub, G. H. and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, pp. 557-558, 1989.

[19] William M. Siebert, "Circuits, Signals, and System", *McGraw-Hill*, 1996.

[20] James K Roberge, "Operational Amplifiers: Theory and Practice", *John Wiley & Sons*, 1975.

[21] Roger T. Howe, and Charles G. Sodini, "Microelectronics: An Integrated Approach", *Prentice Hall*, 1997.

[22] Kevin Skahill, "VHDL for Programmable Logic", *Addison Wesley*, 1996.

Appendix A. VHDL codes for Digital Controller

Top.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity top is
  port (clk, frm, frmclk, we, reset: in std_logic;
        pixin: in std_logic_vector(7 downto 0);
                        refin: in std_logic_vector(7 downto 0);
        pixval: buffer std_logic_vector(7 downto 0);
                        pixval2: buffer std_logic_vector(7 downto 0);
                        pixval3: buffer std_logic_vector(7 downto 0);
                        pixval4: buffer std_logic_vector(7 downto 0);
                        pixval5: buffer std_logic_vector(7 downto 0);
                        refout: out std_logic_vector(7 downto 0);
                        sta: out std_logic_vector(2 downto 0);
                        rowsel: out std_logic_vector (4 downto 0);
                        rowout: out std_logic_vector (2 downto 0));

    attribute pin_numbers of top:entity is
                        "clk:G4 we:A2 frm:A3 frmclk:E23 pixin(7):A4 pixin(6):A5 pixin(5):A6 pixin(4):A7 "&
                        "pixin(3):A8 pixin(2):A9 pixin(1):A10 pixin(0):A11 reset:A12 "&
                        "refout(7):A21 refout(6):A20 refout(5):A19 refout(4):A18 refout(3):A17 refout(2):A15 "&
                        "refout(1):A14 refout(0):A13 "&
                        "rowout(2):A24 rowout(1):A23 rowout(0):A22 "&
                        "pixval(0):A25 pixval(1):B1 pixval(2):C1 pixval(3):D1 pixval(4):E1 "&
                        "pixval(5):F1 pixval(6):G1 pixval(7):H1 "&
                        "pixval2(0):J1 pixval2(1):K1 pixval2(2):L1 pixval2(3):M1 pixval2(4):P1 "&
                        "pixval2(5):R1 pixval2(6):T1 pixval2(7):U1 "&
                        "pixval3(0):V1 pixval3(1):W1 pixval3(2):Y1 pixval3(3):AA1 pixval3(4):AF2 "&
                        "pixval3(5):AF3 pixval3(6):AF4 pixval3(7):AF5 "&
                        "pixval4(0):AF6 pixval4(1):AF7 pixval4(2):AF8 pixval4(3):AF9 pixval4(4):AF10 "&
                        "pixval4(5):AF12 pixval4(6):AF14 pixval4(7):AF15 "&
                        "pixval5(0):AF16 pixval5(1):AF17 pixval5(2):AF18 pixval5(3):AF19 pixval5(4):AF20 "&
                        "pixval5(5):AF21 pixval5(6):AF22 pixval5(7):AF23 "&
                        "refin(7):AF25 refin(6):AF24 refin(5):AE26 refin(4):AD26 refin(3):AC26 "&
                        "refin(2):AB26 refin(1):AA26 refin(0):Y26 "&
                        "sta(2):W26 sta(1):V26 sta(0):U26 "&
                        "rowsel(4):T26 rowsel(3):R26 rowsel(2):N26 rowsel(1):M26 rowsel(0):L26 ";

end top;

architecture vv2 of top is

        constant totalpixno : std_logic_vector(2 downto 0) := "101";
        constant totalcolno : std_logic_vector(2 downto 0) :="101";

        type state is (idle, getmaxamb, setpix, findaddr, storeaddr, swalloff, feedback, feedreset);
        signal p_s, n_s : state := idle;
        signal pixno : std_logic_vector(2 downto 0) := "000";
        signal colno : std_logic_vector(2 downto 0) := "000";
        signal updatepixno, swall, stnoref, alloffst, addrfound : std_logic := '0';
        signal weaddrmem, weaddrmem2, weaddrmem3, weaddrmem4, weaddrmem5 : std_logic := '0';
        signal st, st2, st3, st4, st5 : std_logic := '0';
        signal endecode, enalloffdet, envaluefinder, enaddrfinder, ensteadydet, ensteadydetnoref : std_logic := '0';
```

```
signal tmaxaddr, finderout : std_logic_vector (15 downto 0) := "0000000000000000";
signal pixaddr, pixaddr2, pixaddr3, pixaddr4, pixaddr5 : std_logic_vector (15 downto 0) :=
"0000000000000000";
signal addrfrommem, addrfrommem2, addrfrommem3, addrfrommem4, addrfrommem5 : std_logic_vector
(15 downto 0) := "0000000000000000";
signal amb, tmaxval : std_logic_vector(7 downto 0);
signal tpixval, tpixval2, tpixval3, tpixval4, tpixval5 : std_logic_vector(7 downto 0);
signal memaddrloc : std_logic_vector(2 downto 0);

component automaxfinder
        port (clk, frm, we: in std_logic;
                pixin: in std_logic_vector(7 downto 0);
                maxval: out std_logic_vector(7 downto 0);
                maxaddr: out std_logic_vector(15 downto 0));
end component;

component addrfinder
        port (frm, en: in std_logic;
                addrin: in std_logic_vector(15 downto 0);
                addrfound: out std_logic;
                addrout: out std_logic_vector(15 downto 0));
end component;

component valuefinder
        port (clk, frm, we, en: in std_logic;
                addr: in std_logic_vector(15 downto 0);
                pixin: in std_logic_vector(7 downto 0);
                val: out std_logic_vector(7 downto 0));
end component;

component steadydet
        port (frm, en: in std_logic;
                ambient, ref : in std_logic_vector(7 downto 0);
                val: in std_logic_vector(7 downto 0);
                steady: out std_logic);
end component;

component steadynoref
        port (frm, en: in std_logic;
                val: in std_logic_vector(7 downto 0);
                steady: out std_logic);
end component;

component memaddr
port (wen: in std_logic;
      addrbus: in std_logic_vector(2 downto 0);
      datainbus: in std_logic_vector(15 downto 0);
      dataoutbus: out std_logic_vector(15 downto 0));
end component;

component decoder
port (en,swall: in std_logic;
      pixel_no: in std_logic_vector(2 downto 0);
      rowsel: out std_logic_vector (4 downto 0));
end component;

component alloffdet
port (frm, en: in std_logic;
      ambient: in std_logic_vector(7 downto 0);
      val: in std_logic_vector(7 downto 0);
      steady: out std_logic);
end component;
```

```
begin

        autof : automaxfinder port map (clk, frm, we, pixin, tmaxval, tmaxaddr);
        addrf : addrfinder port map (frmclk, enaddrfinder, tmaxaddr, addrfound, finderout);
        steadnref : steadynoref port map (frmclk, ensteadydetnoref, tmaxval, stnoref);
        decode1 : decoder port map (endecode,swall, pixno, rowsel);
        alloff1 : alloffdet port map (frmclk, enalloffdet, amb, tmaxval, alloffst);

        steadfeed : steadydet port map (frmclk, ensteadydet, amb, refin, tpixval, st);
        steadfeed2 : steadydet port map (frmclk, ensteadydet, amb, refin, tpixval2, st2);
        steadfeed3 : steadydet port map (frmclk, ensteadydet, amb, refin, tpixval3, st3);
        steadfeed4 : steadydet port map (frmclk, ensteadydet, amb, refin, tpixval4, st4);
        steadfeed5 : steadydet port map (frmclk, ensteadydet, amb, refin, tpixval5, st5);

        addrmem : memaddr port map (weaddrmem, memaddrloc, finderout, addrfrommem);
        addrmem2 : memaddr port map (weaddrmem2, memaddrloc, finderout, addrfrommem2);
        addrmem3 : memaddr port map (weaddrmem3, memaddrloc, finderout, addrfrommem3);
        addrmem4 : memaddr port map (weaddrmem4, memaddrloc, finderout, addrfrommem4);
        addrmem5 : memaddr port map (weaddrmem5, memaddrloc, finderout, addrfrommem5);

        valuef : valuefinder port map (clk, frm, we, envaluefinder, pixaddr, pixin, tpixval);
        valuef2 : valuefinder port map (clk, frm, we, envaluefinder, pixaddr2, pixin, tpixval2);
        valuef3 : valuefinder port map (clk, frm, we, envaluefinder, pixaddr3, pixin, tpixval3);
        valuef4 : valuefinder port map (clk, frm, we, envaluefinder, pixaddr4, pixin, tpixval4);
        valuef5 : valuefinder port map (clk, frm, we, envaluefinder, pixaddr5, pixin, tpixval5);

        memaddrloc <= pixno;
        rowout <= pixno;

        sm : process(p_s, reset, addrfound, finderout,stnoref,tmaxval,alloffst,pixno,st,st2,st3,st4,st5,colno)
        begin
                if (reset='1') then
                        n_s <= idle;
           else
                        case p_s is
                                when idle =>
                                        n_s <= getmaxamb;
                                when getmaxamb =>
                                        if (stnoref='1') then
                                                amb <= tmaxval+2;
                                                n_s <= setpix;
                                        end if;
                                when setpix =>
                                        if (tmaxval > "00111000") then
                                                n_s <= findaddr;
                                        end if;
                                when findaddr =>
                                        if (addrfound='1') then
                                                n_s <= storeaddr;
                                   end if;
                                when storeaddr =>
                                        n_s <= swalloff;
                                when swalloff =>
                                        if alloffst='1' then
                                                if (pixno = totalpixno) and (colno = totalcolno) then
                                                        n_s <= feedback;
                                                else
                                                        n_s <= setpix;
                                                end if;
                                        end if;
                                when feedback =>
```

```
                                        if (st='1' and st2='1' and st3='1' and st4='1' and st5='1') then
                                                n_s <= feedreset;
                                        end if;
                                when feedreset =>
                                        if (st='0' and st2='0' and st3='0' and st4='0' and st5='0') then
                                                n_s <= feedback;
                                        end if;
                                when others =>
                        end case;
                end if;
        end process sm;

        translate :
process(p_s,tpixval,tpixval2,tpixval3,tpixval4,tpixval5,refin,addrfrommem,addrfrommem2,addrfrommem3,addrfromm
em4,addrfrommem5,colno)
        begin
                case p_s is
                        when idle =>
                                sta <= "000";
                                pixval <= "11111111";
                                pixval2 <= "11111111";
                                pixval3 <= "11111111";
                                pixval4 <= "11111111";
                                pixval5 <= "11111111";
                                swall <= '1';
                                refout <= "00100000";
                                enaddrfinder <= '0';
                                envaluefinder <= '0';
                                ensteadydet <= '0';
                                ensteadydetnoref <= '0';
                                weaddrmem <= '0';
                                weaddrmem2 <= '0';
                                weaddrmem3 <= '0';
                                weaddrmem4 <= '0';
                                weaddrmem5 <= '0';
                                enalloffdet <= '0';
                                endecode <='0';
                        when getmaxamb =>
                                sta <= "001";
                                swall <= '1';
                                refout <= "00100000";
                                enaddrfinder <= '0';
                                envaluefinder <= '0';
                                ensteadydet <= '0';
                                ensteadydetnoref <= '1';
                                enalloffdet <= '0';
                                endecode <='0';
                        when setpix =>
                                sta <= "010";
                                case colno is
                                        when "001" =>
                                                pixval <= "00000000";
                                        when "010" =>
                                                pixval2 <= "00000000";
                                        when "011" =>
                                                pixval3 <= "00000000";
                                        when "100" =>
                                                pixval4 <= "00000000";
                                        when "101" =>
                                                pixval5 <= "00000000";
                                        when others =>
                                end case;
```

```
		swall <= '0';
		refout <= "00100000";
		enaddrfinder <= '0';
		envaluefinder <= '0';
		ensteadydet <= '0';
		ensteadydetnoref <= '0';
		enalloffdet <= '0';
		endecode <='1';
	when findaddr =>
		sta <= "011";
		swall <= '0';
		refout <= "00100000";
		enaddrfinder <= '1';
		envaluefinder <= '0';
		ensteadydet <= '0';
		ensteadydetnoref <= '0';
		enalloffdet <= '0';
		endecode <='1';
	when storeaddr =>
		sta <= "100";
		swall <= '0';
		refout <= "00100000";
		enaddrfinder <= '1';
		envaluefinder <= '0';
		ensteadydet <= '0';
		ensteadydetnoref <= '0';
		case colno is
			when "001" =>
				weaddrmem <= '1';
			when "010" =>
				weaddrmem2 <= '1';
			when "011" =>
				weaddrmem3 <= '1';
			when "100" =>
				weaddrmem4 <= '1';
			when "101" =>
				weaddrmem5 <= '1';
			when others =>
		end case;
		enalloffdet <= '0';
		endecode <='1';
	when swalloff =>
		sta <= "101";
		pixval <= "11111111";
		pixval2 <= "11111111";
		pixval3 <= "11111111";
		pixval4 <= "11111111";
		pixval5 <= "11111111";
		swall <= '0';
		refout <= "00100000";
		enaddrfinder <= '1';
		envaluefinder <= '0';
		ensteadydet <= '0';
		ensteadydetnoref <= '0';
		weaddrmem <= '0';
		weaddrmem2 <= '0';
		weaddrmem3 <= '0';
		weaddrmem4 <= '0';
		weaddrmem5 <= '0';
		enalloffdet <= '1';
		endecode <='1';
	when feedback =>
```

```
                                        sta <= "111";
                                        pixval <= tpixval;
                                        pixval2 <= tpixval2;
                                        pixval3 <= tpixval3;
                                        pixval4 <= tpixval4;
                                        pixval5 <= tpixval5;
                                        swall <= '0';
                                        refout <= refin;
                                        enaddrfinder <= '0';
                                        envaluefinder <= '1';
                                        pixaddr <= addrfrommem;
                                        pixaddr2 <= addrfrommem2;
                                        pixaddr3 <= addrfrommem3;
                                        pixaddr4 <= addrfrommem4;
                                        pixaddr5 <= addrfrommem5;
                                        ensteadydet <= '1';
                                        ensteadydetnoref <= '0';
                                        enalloffdet <= '0';
                                        endecode <='1';
                                when feedreset =>
                                        sta <= "000";
                                        ensteadydet <= '0';
                                        endecode <='0';
                                when others =>
                        end case;
                end process translate;

                state_clocked : process (clk)
                begin
                        if rising_edge(clk) then
                                if (not(p_s=n_s)) then
                                        p_s <= n_s;
                                        if n_s=feedback then
                                                if pixno=totalpixno then
                                                        pixno<= "001";
                                                else
                                                        pixno<= pixno + 1;
                                                end if;
                                        elsif n_s=setpix then
                                                if pixno=totalpixno then
                                                        pixno<= "001";
                                                        colno<= colno + 1;
                                                else
                                                        pixno<= pixno + 1;
                                                end if;
                                        elsif n_s=idle then
                                                pixno<="000";
                                                colno<="001";
                                        end if;
                                end if;
                        end if;
                end process state_clocked;

end vv2;
```

AddrFinder.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity addrfinder is
  port (frm, en: in std_logic;
                addrin: in std_logic_vector(15 downto 0);
                addrfound: out std_logic;
                addrout: out std_logic_vector(15 downto 0));
end addrfinder;

architecture vv2 of addrfinder is

        signal currentaddr: std_logic_vector (15 downto 0);
        signal count: std_logic_vector (1 downto 0) := "00";

begin

        process(frm)
        begin
                if falling_edge(frm) then
                        if en='1' then
                                if count = "10" then
                                        addrout <= currentaddr;
                                        addrfound <= '1';
                                elsif addrin = currentaddr then
                                        count <= count + 1;
                                else
                                        count <= "00";
                                        currentaddr <= addrin;
                                end if;
                        else
                                addrfound <= '0';
                                addrout <= "0000000000000000";
                                count <= "00";
                        end if;
                end if;
        end process;

end vv2;
```

AllOffDet.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity alloffdet is
  port (frm, en: in std_logic;
                ambient: in std_logic_vector(7 downto 0);
                val: in std_logic_vector(7 downto 0);
                steady: out std_logic);
end alloffdet;

architecture vv3 of alloffdet is

        signal currentval: std_logic_vector (7 downto 0):="00000000";
        signal count: std_logic_vector (3 downto 0) := "0000";

begin

        process(frm)
        begin
                if falling_edge(frm) then
                        if en='1' then
                                if count = "1111" then
                                        steady <= '1';
                                elsif val<=ambient then
                                        count <= count + 1;
                                else
                                        count <= "0000";
                                        currentval <= val;
                                end if;
                        else
                                steady <= '0';
                                count <= "0000";
                        end if;
                end if;
        end process;

end vv3;
```

AutoMaxFinder.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity automaxfinder is
  port (clk, frm, we: in std_logic;
        pixin: in std_logic_vector(7 downto 0);
        maxval: out std_logic_vector(7 downto 0);
        maxaddr: out std_logic_vector(15 downto 0));
end automaxfinder;

architecture vv1 of automaxfinder is

 signal tpixaddr: std_logic_vector(15 downto 0);
 signal tmaxaddr: std_logic_vector(15 downto 0);
 signal tmaxval: std_logic_vector (7 downto 0);
 signal update : std_logic := '0';

begin

 process(clk)
 begin
   if rising_edge(clk) then
     if frm='1' then
            if update='0' then
                    maxaddr <= tmaxaddr;
                    maxval <= tmaxval;
                    update <= '1';
                    tmaxval <= "00000000";
                    tmaxaddr <= "0000000000000000";
                    tpixaddr <= "0000000000000000";
            end if;
     else
            update <= '0';
            if we='0' then
                    tpixaddr <= tpixaddr + 1;
                    if pixin>=tmaxval then
                            tmaxval <= pixin;
                            tmaxaddr <= tpixaddr;
                    end if;
            end if;
    end if;
  end if;
 end process;

end vv1;
```

Decoder.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity decoder is
  port (en,swall: in std_logic;
        pixel_no: in std_logic_vector(2 downto 0);
        rowsel: out std_logic_vector (4 downto 0));
end decoder;

architecture vv3 of decoder is

begin

        rowsel <= "11111" when swall = '1' else
                                "00001" when pixel_no="001" and en='1' else
                                "00010" when pixel_no="010" and en='1' else
                                "00100" when pixel_no="011" and en='1' else
                                "01000" when pixel_no="100" and en='1' else
                                "10000" when pixel_no="101" and en='1' else
                                "00000";

end vv3;
```

MemAddr.vhd

```
library ieee;
use ieee.std_logic_1164.all;
library cypress;
use work.lpmpkg.all;

entity memaddr is
  port (wen: in std_logic;
        addrbus: in std_logic_vector(2 downto 0);
        datainbus: in std_logic_vector(15 downto 0);
        dataoutbus: out std_logic_vector(15 downto 0));
end memaddr;

architecture vv3 of memaddr is

        signal res: std_logic := '0';

begin

U1: mram_dq     -- Single-port RAM with separate data in and data out
        generic map(
                lpm_width           => 16,
                lpm_widthad         => 3,
                lpm_numwords        => 5,               -- optional
                lpm_indata          => LPM_UNREGISTERED,  -- optional
                lpm_address_control => LPM_UNREGISTERED,        -- optional
                lpm_outdata         => LPM_UNREGISTERED, -- optional
                lpm_file            => "",              -- optional
                lpm_hint            => speed            -- optional
        )
        port map(
                data        => datainbus,
                address     => addrbus,
                q           => dataoutbus,
                inclock     => open,        -- optional
                outclock    => open,        -- optional
                we          => wen
        );

end vv3;
```

SteadyDet.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;
library cypress;
use work.lpmpkg.all;

entity steadydet is
  port (frm, en: in std_logic;
        ambient, ref : in std_logic_vector(7 downto 0);
        val: in std_logic_vector(7 downto 0);
        steady: out std_logic);
end steadydet;

architecture vv3 of steadydet is
            signal ival: std_logic_vector (7 downto 0);
            signal count: std_logic_vector (6 downto 0) := "0000000";
            constant final: std_logic_vector (6 downto 0) := "1111111";

begin

U1: Mbuf
            generic map(
                        lpm_width       => 8,
                        lpm_hint        => speed                        -- optional
            )
            port map(
                        data            => val,
                        result                                  => ival
            );

            process(frm)
            begin
                        if falling_edge(frm) then
                                    if en='1' then
                                                if count = final then
                                                            steady <= '1';
                                                elsif (ival=ref or ival=ref-1 or ival=ref+1 or ival<=ambient) then
                                                            count <= count + 1;
                                                else
                                                            count <= "0000000";
                                                end if;
                                    else
                                                steady <= '0';
                                                count <= "0000000";
                                    end if;
                        end if;
            end process;

end vv3;
```

SteadyNoRef.vhd

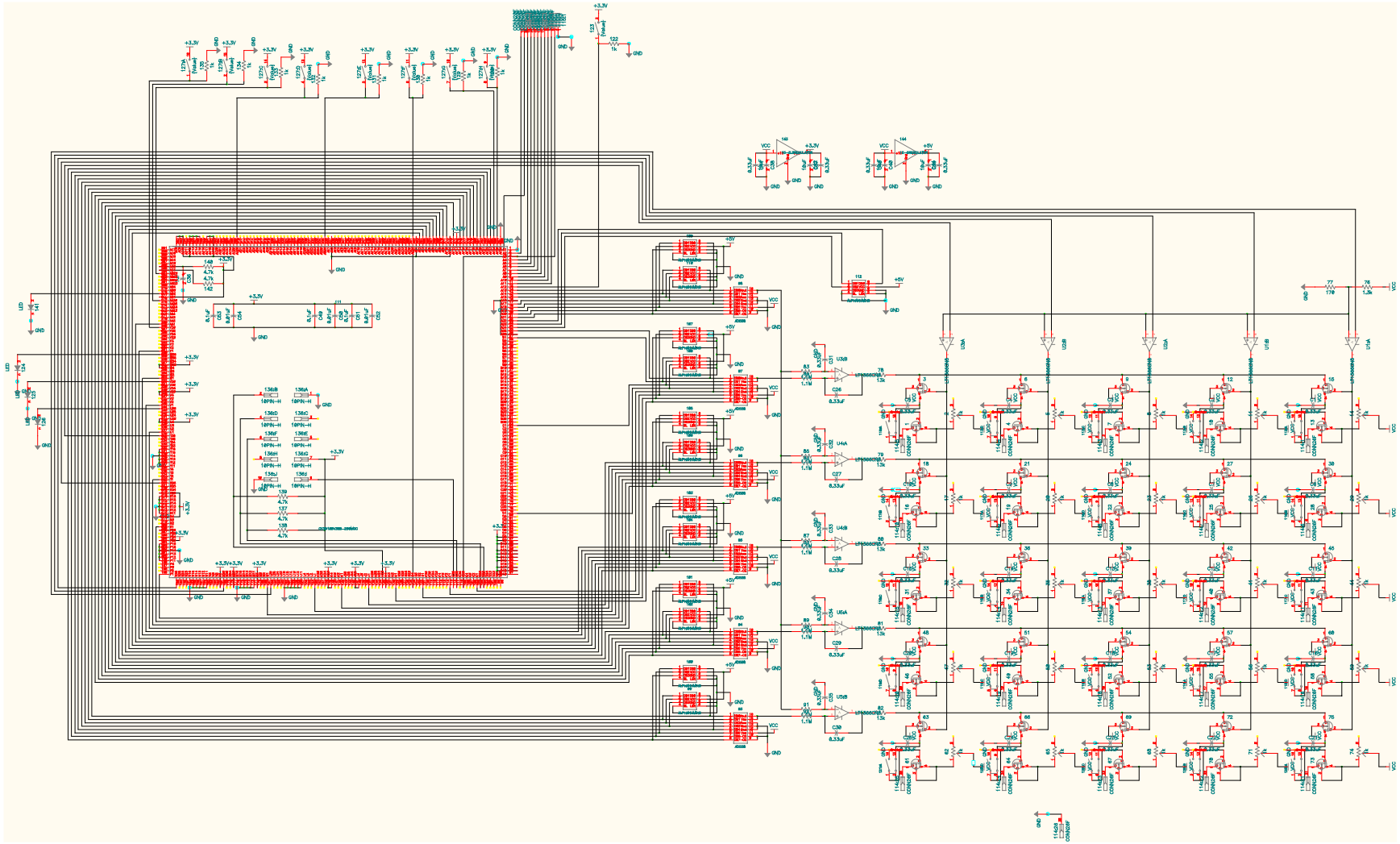
```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity steadynoref is
  port (frm, en: in std_logic;
        val: in std_logic_vector(7 downto 0);
        steady: out std_logic);
end steadynoref;

architecture vv2 of steadynoref is

        signal currentval: std_logic_vector (7 downto 0);
        signal count: std_logic_vector (3 downto 0) := "0000";

begin

        process(frm)
        begin
                if falling_edge(frm) then
                        if en='1' then
                                if count = "1111" then
                                        steady <= '1';
                                elsif val = currentval then
                                        count <= count + 1;
                                else
                                        count <= "0000";
                                        currentval <= val;
                                end if;
                        else
                                steady <= '0';
                                count <= "0000";
                        end if;
                end if;
        end process;

end vv2;
```

ValueFinder.vhd

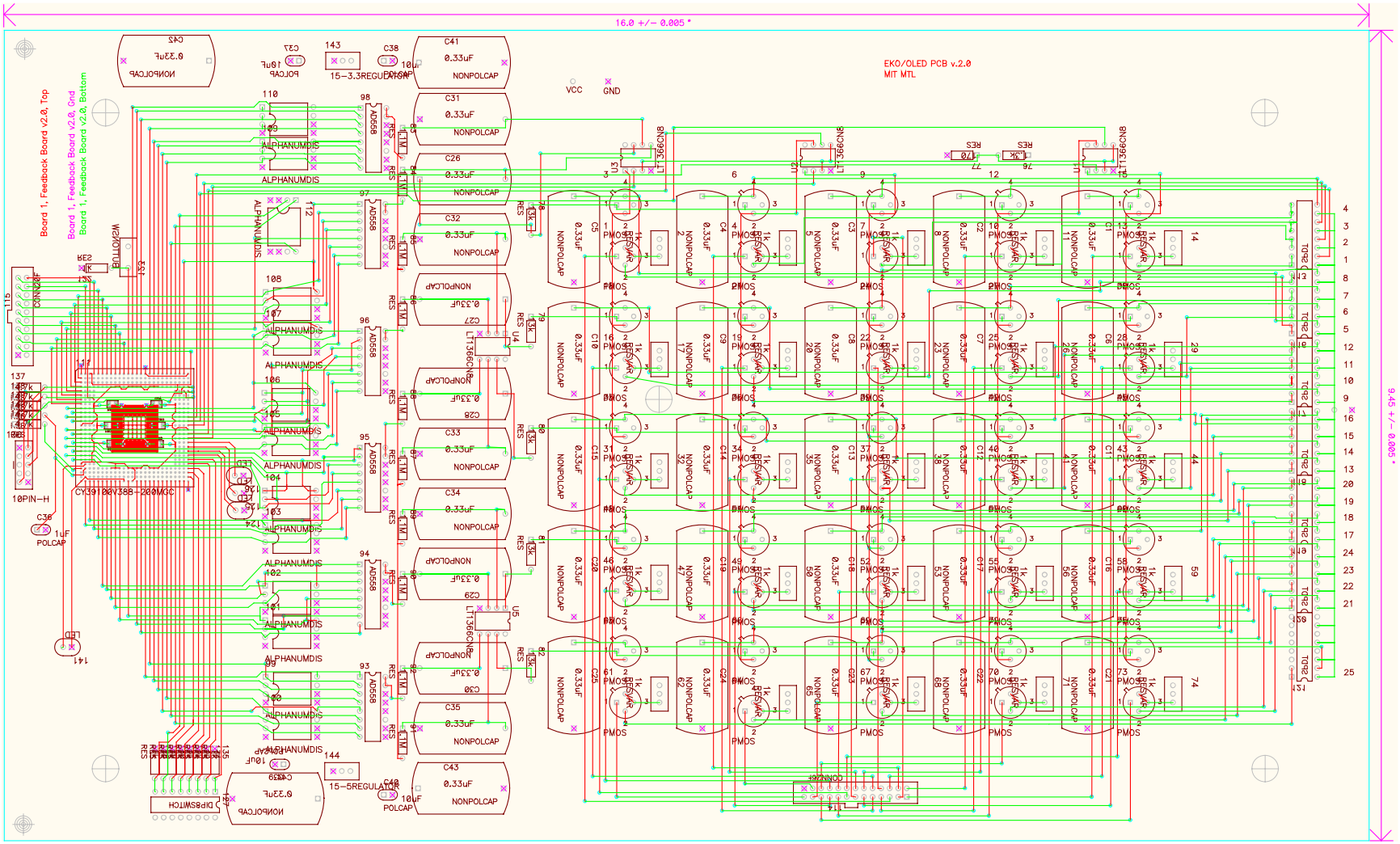
```
library ieee;
use ieee.std_logic_1164.all;
use work.std_arith.all;

entity valuefinder is
  port (clk, frm, we, en: in std_logic;
        addr: in std_logic_vector(15 downto 0);
        pixin: in std_logic_vector(7 downto 0);
        val: out std_logic_vector(7 downto 0));
end valuefinder;

architecture vv1 of valuefinder is

 signal tpixaddr: std_logic_vector(15 downto 0);
 signal tval: std_logic_vector (7 downto 0);
 signal update : std_logic := '0';

begin

 process(clk)
 begin
   if rising_edge(clk) then
     if en='1' then
            if frm='1' then
                        if update='0' then
                                    val <= tval;
                                    update <= '1';
                                    tval <= "00000000";
                                    tpixaddr <= "0000000000000000";
            end if;
            else
                        update <= '0';
                        if we='0' then
                                    tpixaddr <= tpixaddr + 1;
                                    if tpixaddr = addr then
                                                tval <= pixin;
                                    end if;
                        end if;
                        end if;
              else
                val <= "00000000";
              end if;
   end if;
 end process;

end vv1;
```

Appendix B. PCB Design

PCB Schematic



PCB Layout


EKO/OLED PCB v.2.0
MIT MTL
Board 1, Feedback Board v2.0, Top
Board 1, Feedback Board v2.0, Gnd
Board 1, Feedback Board v2.0, Bottom
16.0 +/- 0.005 "
9.45 +/- 0.005 "
VCC
GND
CY39100V388-200MGC
10PIN-H
15-3.3REGULATOR
15-5REGULATOR
DIPSWITCH
ALPHANUMDIS
AD568
LT1366CN8
PMOS
NONPOLCAP
POLCAP
RESVAR
CONN24F